OCTUBRE 86 - 300 ptas.

TodoSpectrum

AÑO II - Número 25

REVISTA EXCLUSIVA PARA USUARIOS

SPECTRUM 128K+2 el nuevo Sinclair

Guía del Hacker: Pentagram

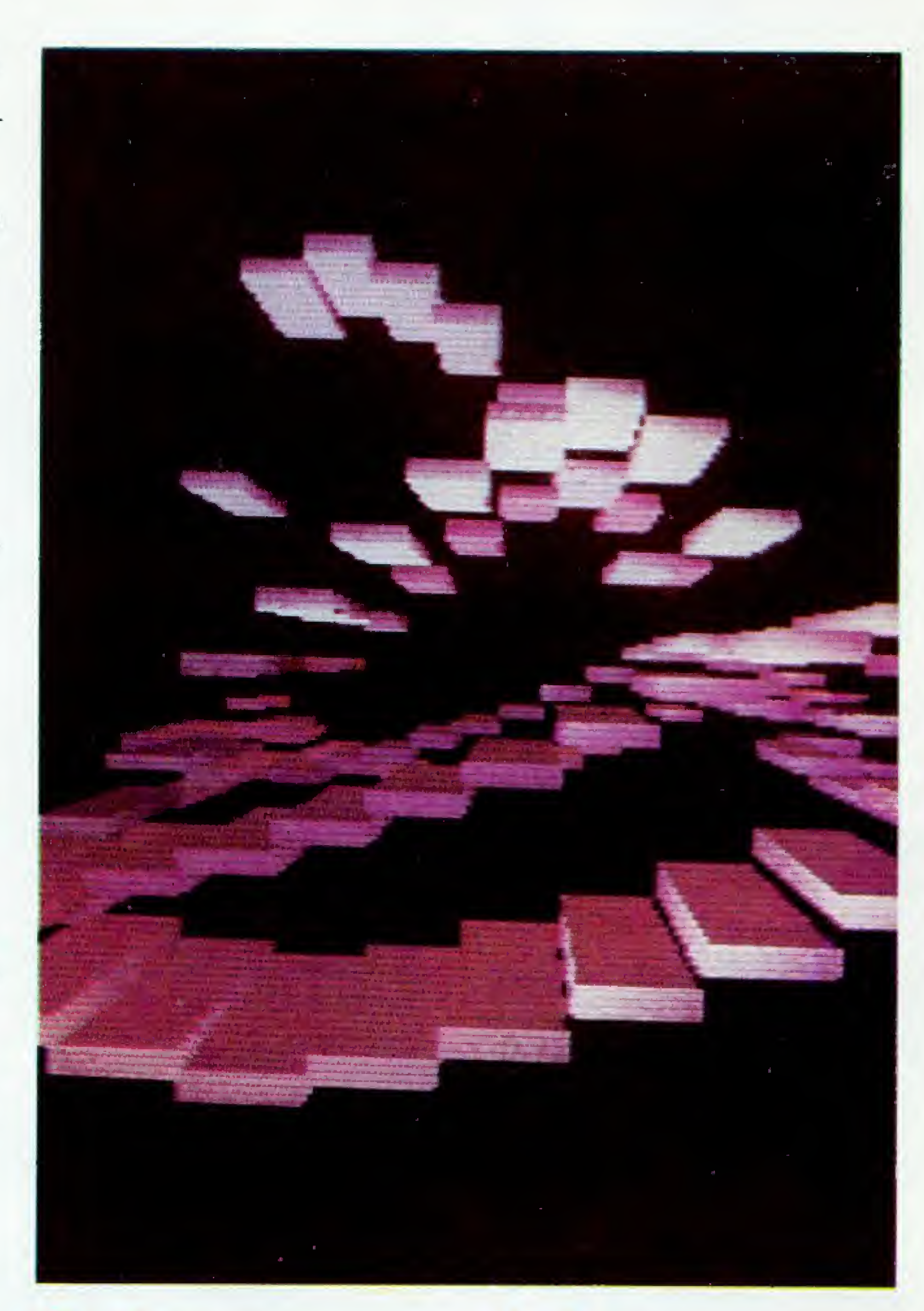
Rotación de UDGs



Interface RS-232-C

SUMARIO

AÑO II - N.° 25 - OCTUBRE 1986



SERVICIO DE EJEMPLARES ATRASADOS



Para hacer su pedido, rellene este cupón **HOY MISMO** y envíelo a:

Todospectrum Bravo Murillo, 377
Tel. 733 96 62 - 28020 MADRID

Ruego me envíen los siguientes ejemplares atrasados de TODOSPECTRUM .... al precio de 300 pts.

El importe lo abonaré

☐ POR CHEQUE ☐ CONTRA REEMBOLSO ☐ CON MI TARJETA DE CREDITO ☐ AMERICAN EXPRESS ☐ VISA ☐ INTERBANK

Numero de mi tarjeta: ....

Fecha de caducidad .... Firma

NOMBRE ....

DIRECCION ....

CIUDAD .... C.P. ....

PROVINCIA ....

Complete su colección de

Todospectrum

A continuación le resumimos el contenido de los ejemplares aparecidos hasta ahora.

Núm. 2 - 300 ptas.
Gráficos profesionales/Desplazamiento pixel a pixel/Utilización de rutinas/Construcción del interface centronics/Programas de utilidad para microdrive/Rutina reset en código máquina/Análisis del editor de textos Tasword/Interfaces para impresoras/Programas.

Núm. 3 - 300 ptas.
Novedades sonimag'84/Ampliando el Basic/Programas para ordenar programas/Gráficos con el VU-3D/Lenguaje Forth/Archivos en microdrive/Programación de un interface de impresora/Programas.

Núm. 4 - 300 ptas.
De profesión: programador/Consola para el Spectrum/Comparación código máquina-Basic/Análisis programa contabilidad/Calendario/Pascal/Programas.

Núm. 5 - 300 ptas.
Floppys para Spectrum/Diseño asistido por ordenador/64 Caracteres por linea/Juego de la vida/Pascal/Asi hacemos las portadas/Control de evaluaciones/Programas.

Núm. 6 - 300 ptas.
Representación de funciones/Todos los caminos conducen a la ROM/Juegos/Pascal/Construcción de un lápiz óptico/Programas de gestión. El SITI/Logo: torgugas para todos/ Interrupciones del Z-80/Programas.

Núm. 7 - 300 ptas.
Del 48 al PLUS paso a paso/¿Plotter para Spectrum?/Juegos/Libros de código máquina/Lápiz óptico. Programación del montaje/El LOGO en la escuela/Pascal/Floppys para Spectrum/Programas.

Núm. 8 - 300 ptas.
Amplia tu memoria... a 48 K/Arquitectura: análisis del PREYME/Juegos/FORTH. Nociones básicas/Una clave, please/QL Magazine. Ultimas novedades, análisis de software, Lenguajes/Aula informática con Spectrum/Programas.

Núm 9 - 300 ptas.
Spectrum parlanchin/Juegos/Aula informática con Spectrum/Análisis: Comercial 4/Pascal/Periféricos: Wafadrive/QL Magazine: EASEL lo mejor de PSION. Música con QL/Desplazamiento Pixel a Pixel, aportación de lectores/Programas/Programer II.

Núm. 10 - 300 ptas.
Discos: invesdisc 200/Juegos/Dos programas simultáneos/Protección del software/Conozca extremadura, consulte a su ordenador/Desensamblador Z-80/Sofware educativo/QL Magazine: novedades Informat, Hoja de cálculo, Ajedrez/Construya su propio Joystick/Pascal/programas.

DISPONEMOS DE TAPAS ESPECIALES PARA SUS EJEMPLARES DE ZX
(sin necesidad de encuademación)

Núm. 11 - 300 ptas.
Actualidad/La otra cara del LOGO/Juegos/El Spectrum habla castellano/SOFTAID ayuda para Etiopia/S.O.S. aqui el Spectrum/Dibujar con lápiz óptico/QL Magazine: Procesador de textos. Teclas de función programables/Programas.

Núm. 12 - 300 ptas.
Actualidad/Inteligencia artificial/Lápiz óptico dk'TRONICS/Juegos/Análisis/Bingo/Z-80 PIO/Código máquina/Análisis: MASTERFILE/Programas.

Núm. 13 - 300 ptas.
Actualidad/Discos: Discovery 1/Juegos/Inteligencia artificial/Un nuevo sistema operativo/QL Magazine: Archive, Cartridge doctor. Aplicaciones comerciales/Código máquina/Programas.

Núm. 14 - 300 ptas.
Actualidad, Spectrum 128/Cálculo de estructuras para ingenieros y arquitectos/HELP utilidades en microdrive/Juegos/El microdrive ese desconocido/Código máquina/QL Magazine: GRAPHIC QL. Juegos. Discos de 720 K/Un nuevo operativo/Programas.

Núm. 15 - 300 ptas.
Actualidad/Spectrum 128/Un nuevo operativo/Circulos redondos/Juegos/Utilidades: BETA-BASIC/QL Magazine: Introducción al SUPER BASIC. Nuevas utilidades/Hardware: Puertas lógicas/Código máquina/Programas.

Núm. 16 - 300 ptas.
Actualidad/Cinco horas con SCREEN$/Hardware práctico/Cálculos de infinita precisión/Juegos/Un nuevo operativo/QL Magazine: Gráficos en SUPER-BASIC. Dibujando con ratón. Archivos con Archive. Programa/La última batalla, Juego estratégico.

Núm. 17 - 300 ptas.
Actualidad/Gráficos interactivos/Juegos/Código máquina/Un nuevo operativo/Trucos de programación/QL Magazine: Radiografia del QL. Gráficos en SUPER-BASIC/Libros/Programmas.

Núm. 18 - 300 ptas.
Actualidad/Introducción al C/Libros/Juegos/De cinta a microcinta/Visión panorámica de los microprocesadores más comunes/QL Magazine: Copy de grises. Microprocesadores 68000, una familia numerosa/Curioseando en la ROM/Programas.

DIRECTOR:
Enrique F. Larreta
REDACTOR JEFE:
Emiliano Juárez
REDACCION:
Ignacio Borrell, Octavio López,
Antonio del Rio
DISEÑO:
Ricardo Segura y Benito Gil

Editado por PUBLINFORMATICA, S. A.
Bravo Murillo, 377. 5.º A. Tel.:
733 74 13 - 28020 Madrid
Presidente:
Fernando Bolín
Director Editorial Revistas de Usuarios:
Juan Arencibia
Director de Ventas:
Antonio González
Producción: Miguel Onieva
Servicio al cliente:
Julia González. Tel.: 733 79 69
Administración:
PUBLINFORMATICA, S. A.
Publicidad Madrid:
Emilio García
Dirección, Publicidad y Administración:
Bravo Murillo, 377. 5.º A. Tel. 733 74 13.
Télex: 48877 OPZX e 28020 Madrid
Publicidad Barcelona:
Lidia Cendrós. Pelayo, 12. Tels. (93)
318 02 89 - 301 47 00, ext. 27 y 28.
08001 Barcelona
Depósito legal: M-29041-1984
Distribuye S.G.E.L. Avda. Valdelaparra,
s/n. Alcobendas (Madrid).
Fotomecánica: Karmat, C/ Pantoja, 10.
Madrid.
Fotocomposición: Espacio y Punto
Imprime: Héroes, C/ Torrelara, 8. Madrid.
Distribuidor en VENEZUELA,
SIPAM, S. A.
AVD. REPUBLICA DOMINICANA, EDIF.
FELTREC - OFICINA 4B BOLEITA SUR
CARACAS (VENEZUELA)
Esta publicación es miembro de la
Asociación de Revistas de
Información ari asociada a la
Federación Internacional de Prensa
Periódica, FIPP.

SUSCRIPCIONES:
Rogamos dirijan toda la correspondencia
relacionada con suscripciones a:
TODOSPECTRUM EDISA: Tel. 415 97 12
C/ López de Hoyos, 141-5º
28002 MADRID
(Para todos los pagos reseñar solamentes
TODOSPECTRUM)
Para la compra de ejemplares atrasados
dirijanse a la propia editorial
TODOSPECTRUM
C/ Bravo Murillo, 377. 5.º A
Tel. 733 74 13 - 28020 MADRID

Si deseas colaborar en TODOSPECTRUM
remite tus artículos o programas a Bravo
Murillo, 377. 5.º A. 28020 Madrid. Los
programas deberán estar grabados en
cassette y los artículos mecanografiados.
A efectos de remuneración, se analiza cada
colaboración aisladamente, estudiando su
complejidad y calidad.

EDITORIAL

EL NUEVO SPECTRUM

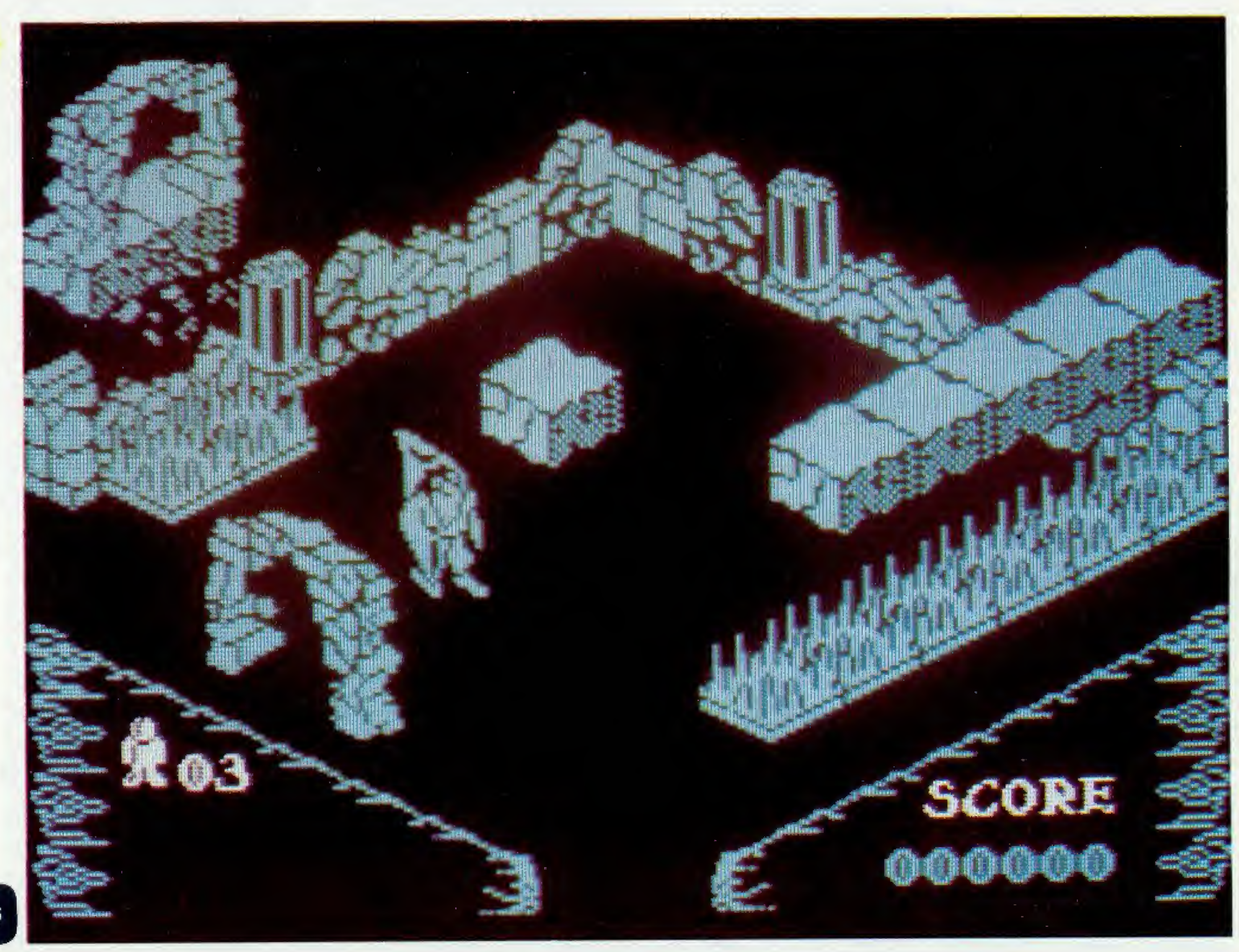
Como estaba previsto, la Personal Computer World Show fue el escenario en que Amstrad presentó el nuevo Spectrum 128K +2. En contra de los rumores que se dejaron oír hace unos meses, no se trata de una máquina revolucionaria, sino más bien de un Spectrum 128 con algunas mejoras y un aspecto diferente. La misma capacidad de memoria, gráficos y sonido, pero por fin un teclado verdaderamente profesional y un cassette incorporado que le confiere un innegable parecido al Amstrad 464. También posee dos ports de joystick, que, por desgracia, sólo podrán utilizarse con los nuevos joysticks Sinclair. Y como ya viene siendo habitual, Amstrad ha vuelto a rizar el rizo ofreciéndolo a un precio inferior al del propio Spectrum 128.

Precisamente la tendencia generalizada a la bajada de precios es la noticia más destacable últimamente en el campo de la microinformática. No hace mucho tiempo (apenas unos meses), comprar un IBM PC o cualquiera de las máquinas compatibles nos hubiera costado más de trescientas mil pesetas. Ahora, por menos de la mitad, es posible adquirir un compatible más rápido, con más memoria y con mejor resolución gráfica que el ordenador de IBM, el Amstrad PC 1512, auténtico centro de atención de la Personal Computer World Show. Y además de Amstrad, Investrónica y Spectravídeo también están ofreciendo compatibles PC a precios de ordenador doméstico, poniendo al alcance de los particulares equipos de elevadas prestaciones que hasta hace poco estaban restringidos al ámbito empresarial.

Pentagram

PENTAGRAM es la última creación de ULTIMATE, y tiene un enorme parecido con los famosos Knight Lore y Alien 8. Sin embargo, aporta algunas características interesantes que lo hacen destacar sobre éstos.

Lo primero que intento hacer a la hora de enfrentarme con un juego para descubrir sus secretos, es conocer lo más posible acerca de él, su argumento, el objetivo, la forma de jugar, etc. Hay veces que esto resulta bastante difícil, pues las instrucciones pueden resultar enigmáticas (los señores de ULTIMATE son especialistas en esto). Sin embargo, consultando diversas fuentes, conseguí enterarme de qué

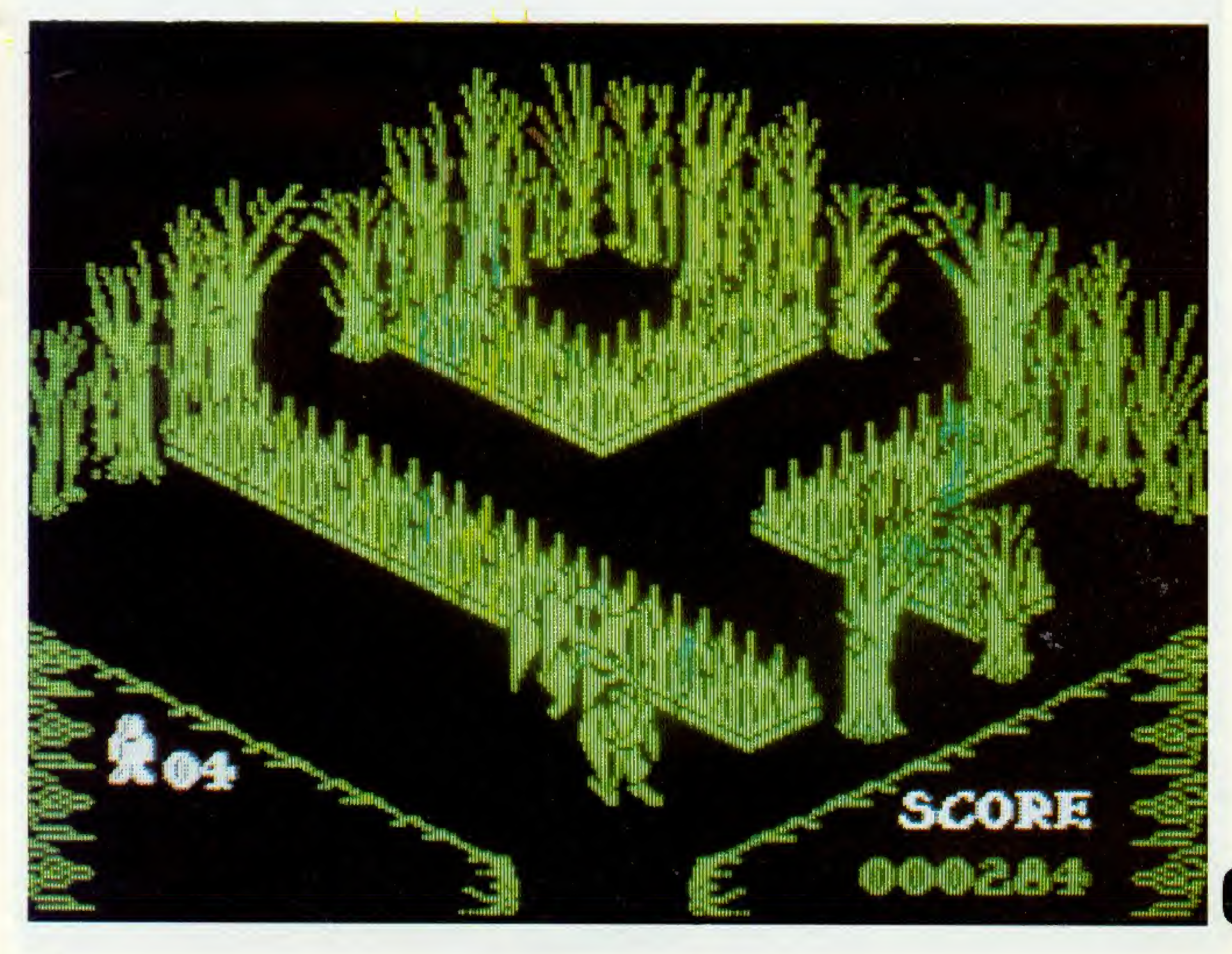


era lo que había que hacer. El objetivo del juego es encontrar el Pentagrama. Para ello, ayudados de unos cubos llenos de agua mágica y que podremos conseguir disparando a los pozos que hay en algunas pantallas, deberemos arreglar cuatro especies de «obeliscos» que se encuentran en otras tantas pantallas. Una vez hecho esto, aparecerá en una habitación, hasta entonces vacía, el Pentagrama, que no es otra cosa que una estrella de cinco puntas inscrita en un círculo. Ya sólo nos queda colocar en cada una de las puntas una de las cinco piedras con extrañas marcas que se encuentran repartidas aleatoriamente por las 139 pantallas que tiene el juego.

Tras enterarme es esto, cargué el programa para hacerme una primera idea acerca de la protección y jugar unas cuantas partidas. Esto último es importante, pues debemos saber exactamente cómo se comporta nuestro personaje en determinadas situaciones, ya que esto nos ayudará a imaginarnos un poco cómo serán las rutinas que lo controlan, lo que nos facilitará su identificación si damos con ellas.

Instalación en el microdrive

La protección no parece compleja en absoluto. Lo primero que carga son dos programas BASIC. El primero dibuja el logotipo de Erbe (la distribuidora del programa para España) y carga el segundo. Aparentemente no tiene ninguna otra función. Probé a saltármelo y cargar directamente el segundo programa BASIC. Al finalizar la carga pude comprobar que el juego funcionaba perfectamente. Ahora sólo tenía que estudiar el segundo programa BASIC. A primera vista sólo tiene una línea, con número cero, que contiene dos pokes típicos de las protecciones: POKE 23613, 0 y POKE 23659,0. Sin embargo, estos pokes no son lo que parecen ser, sino que en realidad son otras direcciones y otros números los que se pokean, estando cambiados los códigos ASCII de estos para despistar. La verdadera función de estos pokes, una vez se ha descubierto cuáles son, es la de hacer aparecer el resto del pro-



```
   1 DATA "DD21E0F9111C023EFF37C
D560530F1DD21004011001B3EFF37CDF
BFADD21A06D119E7A3EFF37CD56122E"
   2 DATA "05D20000F33EC3329EC42
155B3229FC421003C2208CC221CCC218
03D22EACA22DCD621F2EA11F5921157"
   3 DATA "014E01EDB021A06D11005
E019E7AEDB021FF5D22B25C363E2BF92
B2B223D5C2A535C224B5C36B0230E11"
   4 DATA "22595C360D23368023226
15C22635C22655CFBAFCD0116C39512C
D52B9CB4FC21B84CB572068CB5F0FD8"
   5 DATA "C20984CB67C20E843EBFC
D52B9E6012866F3AF3248C43EC93287B
0069621918478BE20032310F9E512AB"
   6 DATA "3D3207C4C52121A734CDC
5AF063CC5CDDAAFC13E7FCD52B9E6072
81CCB47201EE602201631005E21105F"
   7 DATA "21A7343EDD3287B03E073
248C4C3C5AF10D4C1E110BA3EDD3287B
03E07324BC431005EF3C393AFFD1315"
   8 DATA "CB01AEFDCB016E28FA3A0
85CFE303816FE3A3012D6305F87835F1
60021045B195E2356237E123E7E0E2A"
   9 DATA "CD52B9C301B5219D841B0
321A08411048F010300EDB018BFDDE5D
D21004011001B3EFFCDC204DDE110F9"
  10 DATA "1BAD2802CBE13E04CB412
801AFCB4928023E01CB5128023E02CB5
92B023E03FE04281C5FDDE5DD210DB8"
  11 DATA "6FA7CDBDC5DDE11600218
184191919195F1979E6F0B64F3EBDCD5
2B92802CBD93EF7CD52B9E60828135 3"
  12 DATA "02CBD9C308BF040101020
104020102010401010201047A7574737
26A695B5A5045FFC32D842802CB0B1D"
 200 PAPER 0: BORDER 0: INK 7: C
LEAR 28063: RESTORE : LET a=10:
LET b=11: LET c=12: LET d=13: LE
T e=14: LET f=15: LET ad=6e4
 210 PRINT AT 0,0;"LEYENDO LINEA
:": FOR z=1 TO 12: PRINT AT 0,16
;z
 220 READ a$: LET b$=a$(LEN a$-3
 TO ): LET a$=a$( TO LEN a$-4):
IF LEN a$/2<>INT (LEN a$/2) THEN
  PRINT  FLASH 1;AT 0,0;"LINEA I
MPAR EN": STOP
 230 LET w=0: FOR x=1 TO LEN a$
STEP 2: LET v=VAL a$(x)*16+VAL a
$(x+1): LET w=w+v
 240 POKE ad,v: LET ad=ad+1: NEX
T x
 250 LET v=0: FOR x=1 TO 4: LET
v=v*16+VAL b$(x): NEXT x: IF v<>
w THEN  PRINT  FLASH 1;AT 0,0;"E
RROR EN LINEA ": STOP
 260 NEXT z
 270 CLEAR : INPUT "QUIERES GRAB
ARLO EN CINTA? "; LINE A$: IF A$
="S" OR A$="s" THEN  SAVE "CARGA
PENTA" LINE 200
 280 FOR X=23300 TO 23327 STEP 3
: READ A,B: RANDOMIZE A: POKE X,
PEEK 23670: POKE X+1,PEEK 23671:
 POKE X+2,B: NEXT X
 290 PRINT "PON EN MARCHA LA CIN
TA ORIGINAL": RANDOMIZE USR 6E4
 300 DATA 49493,150,A,120,A,72,A
,85,A,87,A,88,A,6,A,17,49534,60,
A,4
```

grama. En las líneas que aparecen, lo primero que encontramos son de nuevo los mismos pokes, pero esta vez de verdad. Tras esto hay una llamada a una rutina en código máquina cuya dirección también es distinta de la que aparece en el listado. Lo que viene después es sólo para despistar y no tiene ninguna utilidad. La rutina en código máquina hace un ldir de un bloque de 206 bytes a la parte alta de la memoria. En estos se encuentra una rutina de LOAD de alta velocidad, aunque bastante semejante a la de la ROM. Con esta rutina se carga la pantalla de presentación y el buffer de la impresora, para después ejecutar lo que se ha cargado en el buffer. Aquí se carga el bloque principal a velocidad normal. Ya tengo todos los datos que necesito saber: el bloque principal se carga a partir de la dirección 24064 y tiene 31390 bytes de longitud. Para asegurarme de que no es necesaria la presencia del programa BASIC en la memoria para que el juego funcione correctamente, pruebo a crear una cabecera falsa y cargar solamente el bloque principal con LOAD "" CODE. Al terminar la carga lo ejecuto con RANDOMIZE USR 24064 y compruebo que funciona perfectamente. Ahora me dispongo a pasarlo a microdrive. En realidad, no es que sea necesario pasar un programa a microdrive para estudiarlo, pero resulta verdaderamente tedioso tener que cargar el programa desde la cinta cada vez que probamos un poke y deseamos continuar la investigación.

Para pasar el programa a microdrive el único problema que tengo es que comienza en una dirección muy baja, pero esto es fácilmente superable, pues queda espacio de sobra por arriba, así que lo que hago es cargarlo en una dirección más alta y transportarlo a su verdadera dirección con un LDIR una vez finalizada la carga. Además, también tengo sitio de sobra para ubicar el MONS, cosa que hago en la dirección 57000. Sin embargo no es posible volver al monitor desde el juego, ya que éste utiliza la memoria donde se encuentra el desensamblador para la construcción de gráficos y algunas otras cosas.

Los primeros pokes

Al empezar a desensamblar me da la impresión de que estoy desensamblando el Knight Lore. Es como si hubieran cogido el listado fuente de éste y hubieran hecho los cambios necesarios hasta conseguir el listado del Pentagram. En la dirección 24064 hay simplemente una instrucción DI y un salto a la dirección 44935 tras haber puesto la pila en 24064. En la dirección 44935 empieza el programa principal.

Casi al principio, en 44967, hay un LD A,5 seguido por un LD (42785),A. No puedo resistirme a probar a cambiarlo antes de continuar. Como cabía esperar, es el número de vidas. Con esto ya podemos elegir el número de vidas con que queremos jugar. Ahora, sabiendo que las vidas se almacenan en 42785, resulta fácil encontrar un poke para vidas infinitas, con sólo buscar dónde es decrementado su valor. Me encuentro que se hace dentro de una subrutina que comienza en 49900, y que es llamada un poco después del punto donde se inicializan las vidas. Esto me da pie para pensar que desde la dirección 44997 en adelante se encuentra el bucle de juego, y todo lo que hay antes se encarga de la presentación y las opciones.

El poke para la inmunidad fue más costoso de encontrar, y no lo conseguí hasta bastante después. Encontré un lugar donde se testeaban los valores de (IX+0) y (IX+1), y si ambos eran cero se saltaba a 44997, donde yo ya sabía que se decrementaban las vidas. Probé a eliminar el salto. El resultado no fue el esperado. Todo empezó con normalidad. Me acerqué a unos pinchos y fui destruido. Parecía que el poke no

tenía ningún efecto. Sin embargo, enseguida me di cuenta de que sí lo tenía. Cuando el personaje «explotó», el programa pareció no enterarse de ello. El resto de los seres que había pululando por la pantalla continuaron haciéndolo normalmente. Todo parecía normal, salvo que el protagonista ya no estaba y no volvía a aparecer. La causa es que el punto modificado no se encarga de comprobar si colisionamos con algo, sino de comprobar si ya hemos terminado de explotar. En cualquier caso todas estas investigaciones me servirían después para ir enterándome de qué era lo que debía hacer si quería visualizar una determinada pantalla y que se moviera todo lo que había en ella. Pude averiguar que el bucle principal de juego siempre que no se saliera de pantalla o nos mataran, se cerraba sobre 45018, porque ésta era la dirección a la que se saltaba cuando la comprobación anteriormente citada era negativa. Después pude encerrar dentro de este bucle otro más pequeño que se repetía para cada uno de los objetos existentes en la pantalla, ya sean piedras, fantasmas, disparos o el propio personaje. Cada objeto ocupa 32 bytes, aunque nuestro personaje ocupa el doble porque es tratado como dos objetos separados, uno de cintura para arriba y otro de cintura para abajo. Esta técnica ya era usada en el Knight Lore, y servía, por ejemplo, para utilizar las mismas piernas en distintos personajes (el Sabreman, el soldado, etc.), con el consiguiente ahorro de memoria en los gráficos.

Ahora tenía más reducida la búsqueda, porque sabía que la comprobación de colisión con los objetos había de estar dentro de este bucle pequeño. Lo que me despistó fue que la comprobación se efectuaba dentro de una subrutina a la que no se llamaba con un call. Dentro de el bucle pequeño, en el que cada vez se va actualizando el registro IX para apuntar a los datos del objeto co-

Tabla 1. POKES

Número de vidas	POKE 44968,0
Vidas infinitas	POKE 49917,175
No imprime vidas	POKE 49310,201
Disparo sin soltar tecla	POKE 49465,0
Disparo automático	POKE 49454,24
	POKE 49465,0
Inmunidad	POKE 50247,24
Altura del disparo	POKE 49534,n
Duración sonido durante juego	POKE 49183,n
Pantallas donde se quiere empezar	POKE 49896,n
	POKE 49897,n
	POKE 49898,n
	POKE 49899,n
Muerte instantánea	POKE 50248,0

Tabla 2. Pokes de disparo (todos los pokes en 49493)

Nada	1
Abol	6
Araña	17
Pincho movible	28
Cepo	30
Cepo manso	31
Patas de Sabreman	32
Fantasma	48
Bloque movible	63
«Pedorretas»	66
Tronco movible	72
Mesa movible	73
Charco	74
Charco con planta	75
Piedra movible	79
Bruja	80
Piedra sube	84
Piedra sube-baja	85
Dragón sube-baja	86
Bloque móvil 1	87
Bloque móvil 2	88
Araña aleatoria	89
Cubo	90
Piedra inamovible	91
Dragón móvil 1	92
Dragón móvil 2	93
Objeto invisible	99
Pozo	120
Bloque movible	121
Runo 1	144
Runo 2	145
Runo 3	146
Runo 4	147
Runo 5	148
Disparo normal	150
Bicho con tentáculos	160
Bola amorfa	164
Zombie	168

Tabla 3. Pokes predefinidos

0 Disparo normal
1 Dispara pozo
2 Dispara tronco movible
3 Dispara piedra sube-baja
4 Dispara piedra móvil 1
5 Dispara piedra móvil 2
6 Dispara árbol
7 Dispara araña
8 dispara alto
9 Dispara bajo

Tabla 4 Posibles ubicaciones de runos

En cada partida se eligen de forma aleatoria 5 lugares entre un total de 20 posibles. Para modificar alguno de estos 20, se deberán hacer los siguientes POKES:

POKE D, Código de la habitación
POKE D+1, Coordenada 1
POKE D+2, Coordenada 2
POKE D+3, Coordenada 3 (Vert)

Cuando D toma uno de los valores siguientes:

53669	53673	53677	53681
53685	53689	53693	53697
53701	53705	53709	53713
53717	53721	53725	53729
53733	53737	53741	53454

rrespondiente (empieza en 42863 y va siendo incrementado en 32) se busca una dirección en una tabla, utilizando como índice de la misma el contenido de (IX+0). Entonces se salta a esta dirección con un JP (HL), asegurándose de que después se retornará a 45086, donde continúa el bucle principal. Esto me hace pensar que (IX+0) contiene un código identificador del tipo de objeto de que se trata, y que para cada objeto hay una parte del bucle que debe ser distinta, y que por eso está colocada en una subrutina. Así, según cada caso, se llama a una subrutina u otra. El problema que tenía ahora era averiguar a qué subrutina correspondía el control de nuestro Sabreman. Por suerte recordé que en una de las rutinas llamadas durante la inicialización se hacía un traslado de bytes desde otra dirección hasta 42863, que es justo el comienzo de la tabla de objetos. Estos no podían ser otra cosa que los datos iniciales del personaje (las coordenadas donde aparecemos en la pantalla, la dirección a la que miramos, etc.). El primer byte de éstos era 32, así que busqué qué subrutina correspondía a dicho código. Esta se encuentra en 50240. En ella, tras una llamada a otra dirección, se testea un bit referido a IX, y realiza un salto condicional según su valor. Es casi seguro que es aquí donde se comprueban las colisiones con otros objetos. Pruebo a hacer incondicional el salto, y consigo la ansiada inmunidad. Si en vez de hacerlo incondicional, lo suprimo, el resultado es que morimos nada más aparecer en la pantalla.

Un Super POKE

La verdad es que una vez encontrados los tres pokes que he explicado en el apartado anterior, no se me ocurría qué otros pokes podían ser interesantes. Durante la búsqueda de aquéllos había encontrado algunos pokes curiosos por los efectos que producían, pero ninguno era realmente útil para facilitar la consecución del juego. Como me parecía poca cosa lo que había descubierto anteriormente, decidí intentar añadirle al juego algo que no tiene y que poseen la mayoría de los juegos en tres dimensiones, incluidos Knight Lore y Alien 8: el control direccional. Supongo que no se lo han puesto porque supone la necesidad de una tecla más, sin embargo a mí me parece un verdadero fastidio porque estaba acostumbrado a controlar así al personaje y no lograba hacerme con el control rotacional. Para comenzar me puse a buscar las rutinas de exploración de teclado y joysticks. Descubrí que todas confluían en 48904, donde se almacena el dato leido en la dirección 42807, de modo que comencé a buscar puntos en los que se hiciera referencia a esta dirección. Encontré tres. El primero era el que ya había visto, en el que es inicializada. En el segundo, nada más leído el dato, hace un and y se queda con el bit de recogida de objetos, así que

Sabiendo que el número de vidas se almacena en la dirección 42785, resulta fácil conseguir un poke de vidas infinitas.

ésta no podía ser la rutina principal de control del movimiento. Por eliminación, debería tratarse del tercer punto de los que encontré. Mi decepción fue mayúscula al comprobar que no era así. En ésta se hacía también un and, aunque ahora el bit correspondiente a la tecla de disparo. La única solución era que se refiriera a la dirección mediante los registros IX o IY. Pero antes de continuar, decidí estudiar un poco lo que parecía ser la rutina de disparo. Esta rutina crea en la tabla un nuevo disparo siempre que no existan ya dos de ellos y que la tecla de disparo no la tengamos pulsada desde antes. con esto consegui otros dos pokes, el de poder disparar



varias veces sin necesidad de soltar la tecla entre un disparo y otro, y el de disparo automático. Pero lo más importante es que encontré el punto en el que se creaba la tabla con los daots del disparo. Aquí pude ver que el código del disparo es 150. Entonces pensé: ¿qué pasaría si cambiara el código del disparo?, ¿haré disparos en forma de roca o algo así? Hice la prueba con algunos valores y los resultados fueron más que sorprendentes. Podemos disparar cualquier cosa: desde un tronco que nos servirá para subirnos encima, hasta una araña que será útil a quien piense que las pantallas son demasiado fáciles, pasando por un árbol que queda muy decorativo en medio de la habitación. Cuando los vi los resultados me puse a probar con todos los códigos. Los resultados más interesantes pueden encontrarse en la tabla dos.

El cargador

Con todo lo que había descubierto anteriormente, me encontraba en disposición de elaborar un cargador con diversas nuevas opciones.



Durante el estudio de la rutina de disparo encontré una subrutina que devolvía la dirección a la que estaba mirando el Sabreman. Sabiendo esto elaboré la rutina de control direccional. El principio es muy simple: calculo la dirección hacia la que queremos dirigirnos y la comparo con la dirección hacia la que va Sabreman. Si son iguales modifico el dato para que el programa crea que se ha pulsado la tecla de avanzar. Si son distintas, cambio el dato para hacer creer al programa que se ha pulsado izquierda o derecha, según haya que girar a la izquierda o a la derecha para pasar de la dirección en la que estamos a la dirección en al que queremos estar.

Naturalmente, como utilizamos cuatro teclas para las cuatro direcciones, necesitamos otra más para la opción de salto. Para este menester he reservado la fila de la A a Enter, además de la tecla 4. Por estas razones, el control direccional no se podrá utilizar con la opción de teclado (podremos jugar con el teclado, pero mediante las teclas del cursor o las del joystick Sinclair), ni con el joystick Sinclair que corresponda a las teclas 1-5, así que si queremos el control direccional deberemos utilizar joystick Kempston, los cursores o el joystick Sinclair correspondiente a las teclas 6-0.

Con todo lo que había aprendido sobre el bucle principal mientras buscaba el poke de la inmunidad, no me fue demasiado difícil encontrar la forma de ir visualizando secuencialmente las 139 pantallas. El único problema fue que éstas no estaban numeradas de 0 a 138, como era de esperar, sino de 0 a 149, quedando enmedio algunos huecos que hacen que se bloquee el ordenador al intentar visualizar pantallas inexistentes.

Por último, pensando en el polifacético poke del disparo, pensé quie sería útil poder variar durante la partida el objeto que disparamos. Para esto he incluído en el cargador, 10 teclas con pokes «definibles», y que podremos seleccionar pulsando en modo pausa una tecla del 0 al 9. Los pokes que queramos asignarles habremos de ponerlos en la línea 300 del listado uno, empezando por la dirección y el contenido del poke para la tecla cero y terminando por los de la tecla 9. En la tabla 3 se pueden ver los pokes que se incluyen en el cargador como ejemplo y que pueden ser modificados sin ningún problema.

Cada objeto, ya sean piedras, fantasmas o disparos, ocupa 32 bytes. Sólo Sobreman ocupa el doble, ya que es tratado como dos objetos separados.

Una vez introducido el cargador pondremos la cinta original. Al finalizar la carga, y si todo ha ido bien, aparecerá el conocido mensaje "c 1982 Sinclair Research ltd.", pero sobre fondo negro y sin borrarse el dibujo de la pantalla de presentación. Ahora podemos introducir otros pokes que no necesiten variar a lo largo de la partida, y que por tanto, no necesiten estar entre los 10 predefinidos, como pueden ser el de vidas infinitas, o alguno así. Cuando hayamos introducido todos los pokes que queramos, ejecutaremos el programa con RANDOMIZE USR 24064. Comprobaréis que el juego de caracteres especial del juego ha desaparecido, y que los mensajes se escriben con el juego de caracteres de la ROM, además de que los espacios se sustituyen por iguales. Esto es porque he necesitado la memoria que ocupaba el juego de caracteres para incluir el cargador con las nuevas opciones, ya que no había otro lugar libre en la memoria.

Si todo ha ido bien, el juego empezará normalmente y lo controlaremos rotacionalmente. Para pasar al control direccional, hay que pulsar la tecla C o la N en modo pausa. Para volver al control rotacional, pulsar también en pausa las teclas V o B. Además, con la Z podremos grabar una pantalla en cinta y con la X o la M abortaremos el juego.

Para visualizar las 139 pantallas hay que pulsar ENTER. Para pasar rápidamente a la siguiente, pulsar Shimbol Shift. Si queremos continuar el juego desde la pantalla que aparezca en ese momento, pulsar M. Para terminar de ver el mapa y comenzar una nueva partida, pulsar SPACE. Hay que tener cuidado con la opción M dentro de la opción de visualización de pantallas, pues podemos aparecer encima de un pincho y perder todas las vidas una tras otra. Para ello, antes de pulsar la M hemos de asegurarnos de dónde vamos a aparecer, lo cual nos será indicado por una explosión que aparece cada vez que pasamos a otra pantalla.

Con todo esto, creo que se facilita bastante la tarea de terminar el juego, aunque aún quedan muchas cosas por descubrir. Espero que alguien se anime a intentarlo.

Pablo Ariza

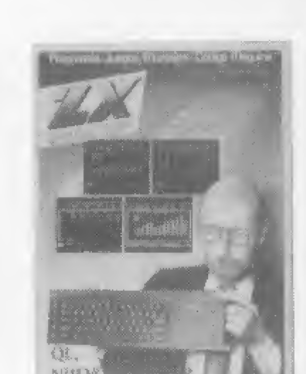


SERVICIO DE

Completa tu colección de ZX.
A continuación te resumimos el contenido de los ejemplares atrasados en existencia.



Núm. 3/300 ptas.
El Spectrum por dentro. Quince programas, juegos y montajes Software.



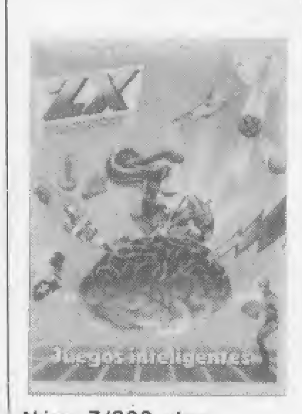
Núm. 4/300 ptas.
QL, el nuevo Sinclair. Dieciocho programas, juegos, montajes, ideas/Novedades.



Núm. 5/300 ptas.
Gráficos y sonido en el Spectrum/Libros/Software/13 programas.



Núm. 6/300 ptas.
Construya su propio juego/13 programas y montajes/ideas/Software.



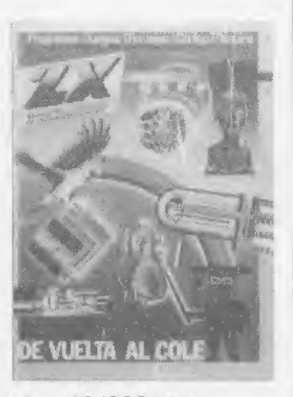
Núm. 7/300 ptas.
Juegos inteligentes/Software/ 11 programas/Libros.



Núm. 8/300 ptas.
La aventura es la aventura/12 programas/Juegos y montajes/Código máquina.



Núm. 9/300 ptas.
Construye tu propio juego. Catorce programas para el verano. Gráficos en el Spectrum.



Núm. 10/300 ptas.
Catorce programas educativos: geografía, cramer, gráficos, razones trigonométricas, elongación. Código máquina.



Núm. 11/300 ptas.
Cómo crear marcianos y otros monstruos.
Diez programas satélites de júpiter, rescate, interés, círculo, préstamo hipotecario.



Núm. 12/300 ptas.
Presentación del Spectrum Plus. Forth, capítulo 1. Gráficos en el Spectrum, 4 parte. Libros. Programas y montajes.



Núm. 13/300 ptas.
Guía del software para el Spectrum todos los programas del mercado. Forth, capítulo 2. Visitamos Sinclair Research. Libros. Programas.



Núm. 14/300 ptas.
Cómo jugar al Hobbit. Gráficos de funciones. Programas de ajedrez. Conexiones con el P I/O. Programas Multiplic, enseñar deleitando. Libros, Forth, tercera parte.



Núm. 15/300 ptas.
Simuladores de vuelo. Forth, cuarta parte. Montajes: Reloj digital para Spectrum. BASIC para principiantes. Libros. Programas.



Núm. 16/300 ptas.
Cassettes: solución a los problemas de grabación. Test de Psicología. Sistema de Desarrollo para el ZX-81. Cinemática. Programas. Animación Gráfica. BASIC para principiantes (2). Forth, quinta parte.

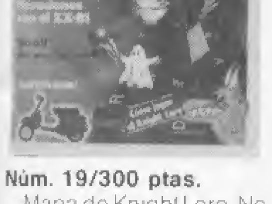


Núm. 17/300 ptas.
Mapa de Atic-Atac. Estira de caracteres. Dinámica de una partícula. Libros. QL Magazine. Programas. Convertidor analógico-digital con el P I/O.

EJEMPLARES ATRASADOS



Núm. 18/300 ptas.
Rentas 85. Forth, sexta parte. Programas. BASIC para principiantes (3). Plotting Gráficos. Libros. Usuarios. Crítica.

Núm. 19/300 ptas.
Mapa de Knight Lore. Noticias. Crítica. Renta 85 (segunda parte). Libros. El ZX-81 aprende a sumar. Scroll de ventanas. Programas. El software que nos invade. BASIC para principiantes (4).

Núm. 20/300 ptas.
Vacaciones con informática. Crítica. Noticias. Programas. Son muy divertidos. Libros. Generación de placas de circuito impreso. Forth. Movimiento armónico simple. Spectrum musical.



Núm. 21/300 ptas.
Mapa de Underwurlde. Noticias. Crítica. ¿Has probado? Programa especial: barquitos. Sois muy divertidos. Libros para el verano. Un poco de física. BASIC para principiantes (5).



Núm. 22/300 ptas.
Noticias. Teclados profesionales. Crítica. ¿Has probado? Programa especial: procesador de textos. Generación de placas de circuito impreso (segunda parte). Programas QL español. Quinielas en Spectrum. BASIC para principiantes (6).



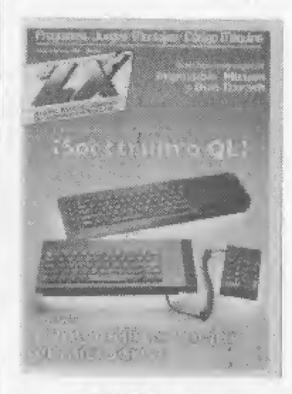
Núm. 23/300 ptas.
Crítica. ¿Has probado? Profanation profanado. Noticias. Discos para Spectrum. Dossier educación: Spectrum en el aula, autoevaluación, Logo. Código máquina. Programación especial: quinielas. Montaje a cámara lenta. BASIC para principiantes (7).



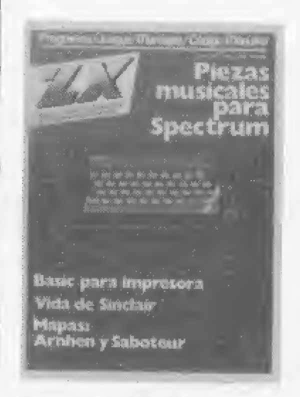
Núm. 24/300 ptas.
Juegos/Mapas del Nodes of Yesod y Lords of Midnight/¿Has probado?/ Sois muy divertidos/Usuario/Ajuste de gráficas/Multisearch/Programas/Montaje: inversor de video para ZX 81/Dossier QL.



Núm. 25/300 ptas.
Juegos/Especial juegos, Mapas y trucos de: Highway encounter, Tir Na Nog, Nightshade/¿Qué es el Stack?/Programa especial/ Código máquina/Lotería primitiva/Standares de la informática/Programas.



Núm. 26/300 ptas.
Spectrum o QL, invasión de los 128/¿Cómo utilizar mejor el microdrive?/Juegos/Mapa del Dun Darach y misión imposible/Programación estructurada/BASIC.



Núm. 27/300 ptas.
La vida de Sinclair/Piezas musicales para Spectrum/Juegos/Mapas del ARNHCM y SABOTEUR/ Areas/BASIC para impresora/El área de variable y la instrucción RST 16.

DISPONEMOS DE TAPAS ESPECIALES PARA SUS EJEMPLARES DE ZX (sin necesidad de encuadernación)



PRECIO UNIDAD **650** ptas.

(en cada tomo se pueden encuadernar 6 números)

Para hacer tu pedido, rellena el cupón adjunto, córtalo y envíalo HOY MISMO a:

ZX, Bravo Murillo, 377 • 28020-MADRID • Tel. 733 74 13

Los ejemplares atrasados de ZX serán una fuente constante de conocimientos, ideas, soluciones y entretenimientos para el futuro. Todo lo anterior hace recomendable que los guardes ordenadamente en una de las tapas especiales para ZX. Cada tapa puede contener 6 ejemplares y cuesta solamente 650 ptas.

Ruego me envíen los siguientes ejemplares atrasados de ZX .... al precio de 300 ptas. cada uno

Por favor envíen .... tapa(s) al precio de 650 ptas. cada una (+ gastos de envío).

El importe lo abonaré:
☐ contra reembolso ☐ cheque adjunto ☐ con mi tarjeta de crédito
☐ American Expres ☐ Visa ☐ Interbank

Fecha de caducidad ....

Número de mi tarjeta ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

NOMBRE ....
DIRECCION ....
POBLACION .... C.P. ....
PROVINCIA ....

INTERFACE SERIE 



El primer programa se encarga del envío de una palabra, y el segundo de la recepción. A ambos se accede mediante instrucciones DEF FN, como se indica a continuación:

```
DEF FN E(a)=USR 65024
DEF FN R()=USR 65081
```

La instrucción FN E(x) tendrá el efecto de enviar por la línea RS-232 el valor de x. Si lo usáramos en una sentencia de asignación –LET a= FN E(x)–, podríamos saber si la orden se ha llevado a cabo, o por contra la hemos interrumpido desde teclado usando la tecla SPACE. Para ello bastará con comprobar si el valor de retorno es mayor que 256. Si así fuera, se ha producido un error.

Asimismo, un LET a=FN R(x) asigna a la variable A el número recibido por la línea. La condición es la ya comentada.

En cuanto a los programas en sí, hay poco que decir que sea novedoso o que no pueda leerse en el listado. En la rutina de emisión, hemos de recoger el dato que se nos envía a través de un DEF FN. Para ello emplearemos una técnica que ya hemos usado en otras ocasiones. El contenido de DEFADD nos da la dirección donde se almacena el argumento del DEF FN que se está procesando –el que nos interesa evidentemente–. A partir de esta dirección (en el área de variables), recogemos el dato de forma simple ya que el número se almacena con formato de entero.

Por otro lado, recomendamos a los interesados en programación ensamblador que examinen con detalle los bucles de las rutinas de entrada y salida. Seguramente les será de utilidad.

En la rutina de recepción se pueden detectar errores, producidos generalmente por desajuste de las velocidades de transmisión. El error tiene lugar si el bit de arranque tuviera una longitud menor de la esperada. Esto significaría que se están transmitiendo datos a más velocidad de la esperada.

Acaso alguien se halla preguntando por la razón de ubicar este programa al final de la RAM. Igualmente, los usuarios de Spectrum 16K habrán exclamado: «Maldita sea mi suerte». Es necesario hacerlo ya que, como muchos sabrán, el acceso a los 16 primeros kilobytes de RAM sufre interrupciones por parte de la ULA, y esto puede hacer inservibles los esfuerzos por conseguir una temporización exacta.

Descripción del Hardware

El interface que presentamos ha sido realizado siguiendo la más pura filosofía Sinclair, intentando conseguir las máximas prestaciones manteniendo costes al mínimo.

Trataremos de ser breves en la descripción del hardware.

El interface está direccionado en el puerto de entrada-salida BFh (1011 1111). Cuando leemos algún dato de este puerto –por ejemplo, mediante la instrucción IN A,( )–, se activan los buffer triestado de IC2 por efecto de IC1b. El resultado es que el Z-80 lee en sus dos bits menos significativos el estado de las líneas TXD y DTR. Los transistores invierten los niveles, pero como el estado bajo corresponde a un uno, y al alto al cero, los datos leidos son los adecuados.

Si accedemos al puerto mencionado en una operación de escritura (OUT ( ),<reg>), la salida de la puerta IC1a se pone en alto y los datos presentes en el bus se cargan en los biestables IC3. Se ha previsto que al encenderse el aparato, automáticamente las salidas se pongan a cero, pasando de este modo al estado inactivo.

Para conseguir tensiones negativas hemos usado dos típicas células diodo-condensador, en lugar de las tradicionales bobinas o transformadores, de difícil obtención o fabricación. Dado que presentan una impedancia de salida relativamente alta, ha sido necesario cargarlas con una corriente aproximadamente constante.

De aquí el valor inusualmente bajo de la resistencia de colector de T3 y T4.

Las resistencias R17 y R18 protegen las salidas frente a cortocircuitos accidentales. La protección incondicional de las salidas sólo puede garantizarse haciendo uso de transistores de más potencia para T2, T3, T5 y T6, tales como el BD 139 y BD 140. No recomendamos la opción, a cambio de que se preste un mínimo de atención al realizar las conexiones.

Aunque se ha incluído tanto en el esquema como en el circuito impreso, generalmente la resistencia entre DSR y masa no se conectará. Realmente, es muy poco útil y disipa gran cantidad de calor.



El conector de salida coincide con el del Interface I.
Indicamos a continuación su patillaje:

5 4 3 2 1 o o o o o o o o o 9 8 7 6		Conector RS-232-C
	1: No Conectado (NC)	
	2: TXD (Entrada)	2
	3: RXD (Salida)	3
	4: DTR (Entrada)	20
	5: CTS (Salida)	5
	6: NC	
	7: GND (Tierra)	7
	8: NC	
	9: 89V (DSR gralmente)	(6)

```
100 DATA 243,221,42,11,92,221,86,4
110 DATA 14,191,237,120,230,1,40,11
120 DATA 62,127,219,254,31,56,243
130 DATA 6,255,251,201,6,9,122,183
140 DATA 23,31,203,195,203,19,237,89
150 DATA 205,117,254,16,244,22,3,237
160 DATA 81,205,117,254,205,117,254,72
170 DATA 251,201,243,14,191,6,1,237
180 DATA 65,237,120,230,2,40,15,62
190 DATA 127,219,254,31,56,243,6,3
200 DATA 237,65,6,255,251,201,205,143
210 DATA 254,237,120,230,2,32,239,6
220 DATA 128,205,117,254,237,120,31,31
230 DATA 203,24,0,48,244,22,3,237
240 DATA 81,72,6,0,251,201,253,102
250 DATA 71,46,6,0,0,0,45,194,122
260 DATA 254,37,0,32,243,0,46,5
270 DATA 45,32,253,0,0,35,201,253
280 DATA 102,71,46,3,0,0,45,194
290 DATA 148,254,37,0,32,244,0,38
300 DATA 2,37,0,32,252,201
310 REM
320 sum=0
330 FOR i=65024 TO 65189
340 READ a
350 sum=sum+a
360 POKE i,a
370 NEXT i
380 IF sum=19057 THEN STOP
390 PRINT "Error en el DATA. Repaselo"
400 REM
410 REM      Ejemplo
420 REM
430 DEF FN E(a)=USR 65024
440 DEF FN R()=USR 65081
450 INPUT "Velocidad de transmision:";vt
460 POKE 23681,INT(19200/vt-0.5)
470 REM      Envio de un caracter
480 LET a$="A"
490 LET b=FN E(CODE a$)
500 IF b>255 THEN PRINT "ERROR de transmision":STOP
510 PRINT "Caracter enviado"
520 REM      Recepcion de un caracter
530 LET a=FN r()
540 IF a>255 THEN PRINT "ERROR: velocidad desajustada":STOP
550 PRINT "Valor recibido:";a;
560 IF a>32 THEN PRINT "(";CHR$ a;")";
570 PRINT
```

```
;*************************
;**  RUTINA DE EMISION  **
;*************************
;(C) Luis Miguel BRUGAROLAS
RS232   EQU   191
        ORG   OFEOOH
EMIRUT  DI
        LD    IX,(5COBH)
        LD    D,(IX+04);Cogemos
;             dato a enviar
        LD    C,RS232;Direccion
;             del puerto
DTWT    IN    A,(C);Lee puerto
        AND   01; DTR
        JR    Z,EMIT
        LD    A,7FH;Muestreamos
        IN    A,(OFEH); SPACE
        RRA
        JR    C,DTWT
        LD    B,OFFH; Indicamos
        EI    ; condicion de no
        RET   ; ejecutado con
;             byte alto a cero.
EMIT    LD    B,09;Num de bucles
        LD    A,D;Dato a enviar
        OR    A;CF = 0
        RLA
EMTLP   RRA
        SET   0,E;CTS en bajo
        RL    E;El bit menos signi
;             ficativo de E es el dato
;             a enviar.
        OUT   (C),E;Lo enviamos
;             con CTS en bajo.
        CALL  DELAY
        DJNZ  EMTLP
        LD    D,3
        OUT   (C),D
        CALL  DELAY
        CALL  DELAY
        LD    C,B; B = 00
        EI
        RET   ;Vuelve con BC=00
;***************************
;**  RUTINA DE RECEPCION  **
;***************************
RECRUT  DI
        LD    C,RS232
        LD    B,01;CTS en alto,
;             RXD en bajo.
        OUT   (C),B;Estamos prepar
WTDT    IN    A,(C)
        AND   02; TXD
        JR    Z,RD;Se recibe bit
;             de comienzo.
        LD    A,7FH
        IN    A,(OFEH)
        RRA
        JR    C,WTDT
ERROR   LD    B,03
        OUT   (C),B;CTS y RXD a O
        LD    B,OFFH;Indicamos
;             error.
        EI
        RET
RD      CALL  DEL.5;Esperamos
;             medio periodo.
        IN    A,(C)
        AND   02;TXD
        JR    NZ,ERROR;!Hay un
;             cero en la linea!
        LD    B,10000000B
GETLP   CALL  DELAY
        IN    A,(C)
        RRA
        RRA   ;CF: bit recibido
        RR    B;Pasa a b7 de B
        NOP   ;Hace tiempo para
;             compatibilizar con
;             DELAY
        JR    NC,GETLP
        LD    D,03;  RXD y CTS a
        OUT   (C),D; cero
        LD    C,B
        LD    B,00
        EI
        RET   ;BC: dato recibido
;*************************
;**  RUTINAS DE RETARDO  **
;*************************
DELAY   LD    H,(IY+71)
DELLP2  LD    L,6
DELLP1  DEFB  0,0,0
        DEC   L
        JP    NZ,DELLP1
        DEC   H
        NOP
        JR    NZ,DELLP2
        NOP
        LD    L,5
DELLP3  DEC   L
        JR    NZ,DELLP3
        NOP
        NOP
        INC   HL
        RET
DEL.5   LD    H,(IY+71)
DEL.52  LD    L,3
DEL.51  DEFB  0,0
        DEC   L
        JP    NZ,DEL.51
        DEC   H
        NOP
        JR    NZ,DEL.52
        NOP
        LD    H,2
DEL     DEC   H
        NOP
        JR    NZ,DEL
        RET
```



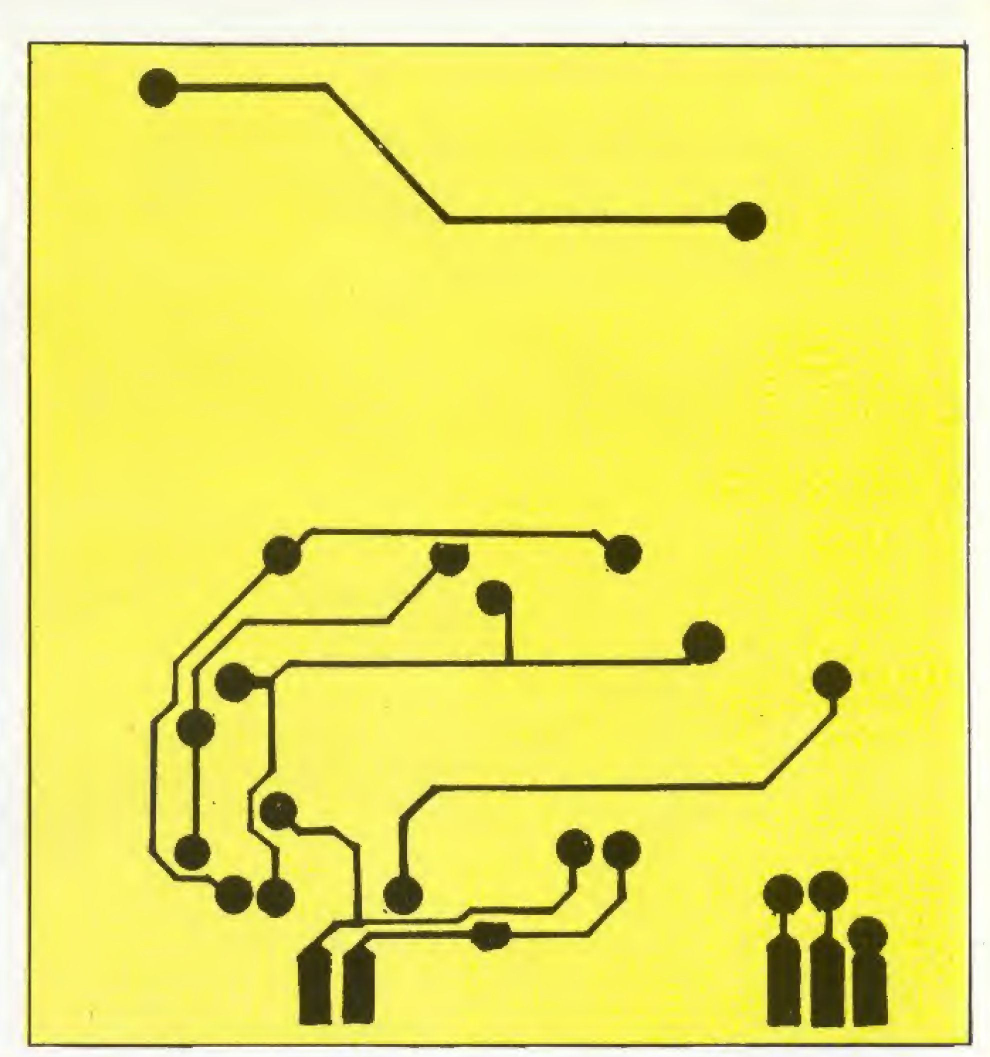
RESISTENCIAS (1/4 W, 5%):
R1, 2 = 2K
R3, 4 = 1K
R5 = 10 Ohm
R6, 7 = 4K7
R8, 9 = 100K
R10, 11 = 33K
R12, 13 = 330 Ohm 1/2 W
R14 = 330 Ohm 1W
R15, 16 = 15K
R17, 18, 19, 20 = 47 Ohm
R21, 22 = 1K
R23, 24, 25 = 10K

TRANSISTORES:
T1 = BC547 (BC548,9)
T2 = BC557 (BC558,9)
T3, 4, 7, 8 = SC159B
T5, 6, 9, 10 = SC149B

CONDENSADORES
C1 = 4,7 nF
C2 = 100 nF
C3, 4 = 470 µF 16V
C5 = 470 µF 25V
C6, 7 = 100 nF

DIODOS:
D1, 2, 3, 4, 5 = 1N4148
D6, 7 = Zener 7,5V 1/4W

CIRCUITOS INTEGRADOS:
IC1 = 74LS27
IC2 = 74LS126
IC3 = 74LS74
IC4 = 555



IORQ
WR
A6
RD
DΦ
D1
IC1a
IC1b
74LS27
IC3a
IC3b
74LS74
IC2a
IC2b
IC2c
74LS126
555
IC4
R1
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19
R20
R21
R22
R23
D1
D2
D3
D4
D5
D6
D7
C1
C2
C3
C4
C6
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
5V
12V
-6V
RXD
CTS
DTR
TXD
DSR

IC1
IC3
IC2
IC4
R 23
R24
R25
C5
C7

JUEGOS

COBRA'S ARC

DINAMIC

SPECTRUM 48 K



Los sacerdotes de una cultura milenaria vivian plácidamente en un reino lejano, dedicados por entero a la veneración del dios Cobra. En su honor construyeron un templo inmenso en el que durante siglos fueron depositando tesoros, joyas, oro... Asustados por las profecías del adivino Werdhal, que vaticinaba un maremoto que asolaria sus tierras, construyeron alrededor del tesoro un inmenso barco, tan grande como una pirámide. Un inmenso barco que quedo perdido entre las gigantescas olas cuando la profecía se cumplió; lejos, muy lejos, nadie sabe dónde.
Los aficionados a los juegos de aventura «genuinos» estamos de enhorabuena. No es cosa corriente el que se produzcan este tipo de juegos por y para hispanoparlantes, y la verdad es que ya estábamos hartos de andar de aquí para allá con el diccionario intentando acoplarnos al molesto «argot anglosajón» que suelen utilizar. Una vez más ha sido Dinamic quien ha roto el muro que parecía impedir a las casas españolas el sacar a la luz juegos de este tipo. Pero no por ser una excepción va a librarse este juego de la implacable crítica que es común en estas páginas, y ya va siendo hora de que nos internemos en el juego para ver qué puede tener de positivo y qué de menos positivo.
Junto a la buena presentación y cuidado por el detalle (algo, por otra parte, difícilmente separable del nombre de Dinamic), debe resaltarse el uso que se hace de un menú de iconos como forma de dirigir al personaje en lugar del clásico diálogo, más complejo, pero seguramente preferido por todo aventurero que se precie. De todas formas no deja de ser algo bastante «al día» que es agradecible como forma de experimentar nuevas vías en este mercado. Además cada mensaje de los que nos da el ordenador es acompañado por su correspondiente frase (sí, habla, pero de forma bastante «gangosa» y a un nivel bajísimo).
Cuando nos adentramos algo en el juego nos encontramos con que deja bastante que desear a quien esté acostumbrado a los grandes juegos de aventura ingleses como son el Hobbit o tantos otros. La flexibilidad de movimientos, personajes y diálogos tan patente en estos no se encuentra fácilmente cuando tenemos que limitarnos a un pequeño juego de iconos y seguir un flujo demasiado rígido para conseguir permanecer vivos algunos minutos.



FRIDAY THE 13TH

DOMARK

SPECTRUM 48 K



Basado en la conocida película de la Paramount, Viernes 13, es este un juego que se ha dado a conocer de tal forma por la estudiada campaña de marketing que le ha precedido, que hoy hablan de él incluso quienes jamás han llegado a probarlo. Comienza a ser esto algo normal, derivado del increíble crecimiento del mercado en los últimos tiempos y su gran dinamismo. Un juego hoy sólo está en cartel durante dos o tres meses, y en ese tiempo no da tiempo a que corra la voz de la crítica popular como en los primeros tiempos; aquellos tiempos heroicos en los que sólo nos decidíamos a gastar nuestro dinero en un juego cuando era de dominio público que éste era una obra de arte. Hoy las cosas han cambiado, los intereses económicos son muy fuertes y no son ya programadores los que controlan las empresas de software, sino especialistas en publicidad y psicología de masas.

Pero hay que recordar que bajo cada título hay un programa para nuestro ordenador, un programa que es lo que va (o no) a hacernos disfrutar y lo que se supone que deberíamos estar comentando en esta sección.



Un tranquilo día de campo en el lago de cristal va a ser turbado cuando Jason, un zombie en toda regla, comienza a atacar a tus amigos y compañeros de acampada. Deberás buscarlo entre ellos y luchar con él para vencerle. Recuerda que sólo hay un lugar donde Jason no se atreve a entrar; allí donde se encuentre la cruz podrá ser el santuario donde se refugien tus amigos. Valor y adelante.

Como podemos ver, la trama inicial (por otra parte condicionada por la película) es bastante clásica y no aporta apenas nada realmente interesante, pero lo más significativo es que tampoco descubramos nada innovador cuando nos fijamos en la parte técnica del asunto. Unos sprites y gráficos en general bastante «sosos», dos dimensiones, un mapa más bien pequeño,... en fin, que pasó por nuestros ojos sin llamarnos la atención en absoluto. Tampoco es que tenga grandes defectos ni faltas graves, pero hoy en día es necesario algo más que un carecer de defectos para triunfar en un mercado tan saturado de títulos.

WINTER GAMES

EPYX

SPECTRUM 48 K

Aunque parece que ha pasado algo la fiebre de los juegos deportivos que aquél «Decathlon» puso tan de moda, aún siguen vendiéndose «versiones» más o menos diferenciadas entre sí que casi permiten hablar de una importante subclase de juegos; juegos que han marcado un hito en la «movida» evolución de este interesante mercado en los últimos años.

Pero es esta vez un juego que aporta algo nuevo a esta repetitiva clase. Como indica su propio nombre se trata de unos Juegos de Invierno, con sus pruebas características que ponen las cosas interesantes desde el comienzo.

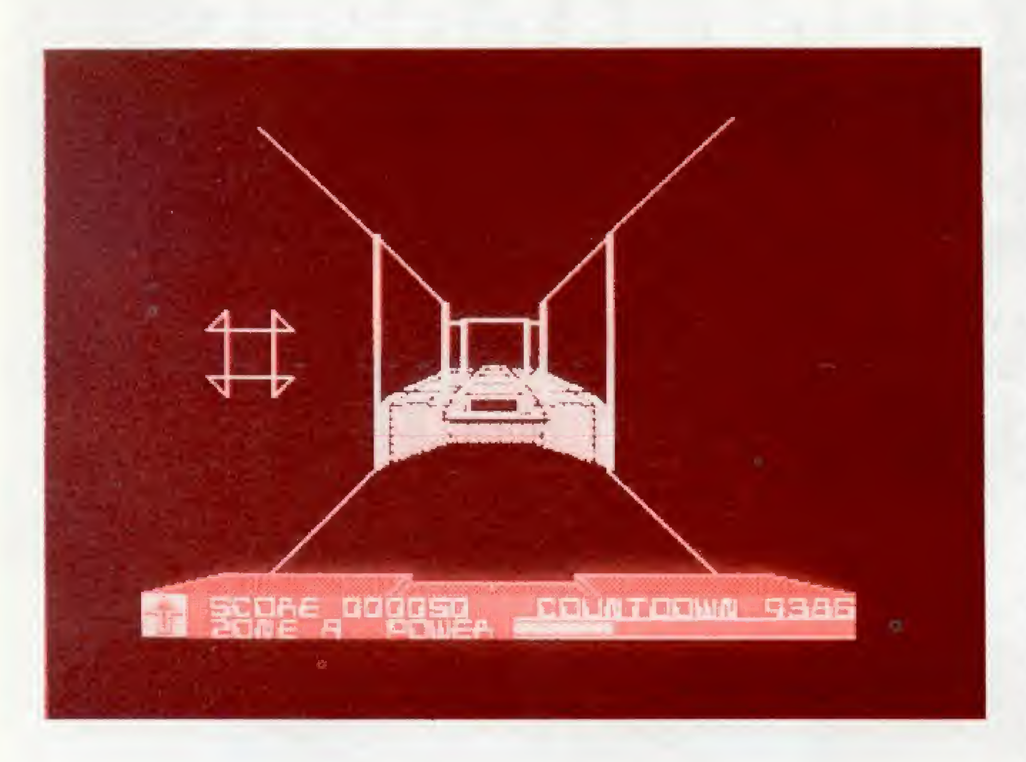
Preparémonos para competir sobre la nieve y el hielo en las más duras pruebas a las que puede verse sometido un curtido deportista invernal.

Bobsled, salto de ski, patinaje artístico, patinaje estilo libre, Hot Dog aéreo y ski de fondo son las competiciones en las que deberemos probar nuestra pericia. Pruebas entretenidas y bien acabadas en general que hacen de este juego (Decathlon aparte) todo un «cabeza de lista» dentro de su gama. Típico juego de competición, sólo resulta realmente divertido cuando son muchos los jugadores que se turnan en busca del oro (admite hasta ocho posibles).

En la parte técnica cabe resaltar unos fondos muy espectaculares junto a un personaje principal demasiado sencillo pero bien animado. La sensación es buena y «se pega a los dedos» con facilidad. En presentación se cumple a la perfección, haciéndolo un juego muy completo dentro del ámbito que le corresponde. El estar dividido en dos partes (tres pruebas por cara, al viejo estilo) permite a los programadores derrochar más memoria y trabajar más a gusto, aunque, como parte negativa, el jugador perderá bastante tiempo con las tediosas cargas del cassette.



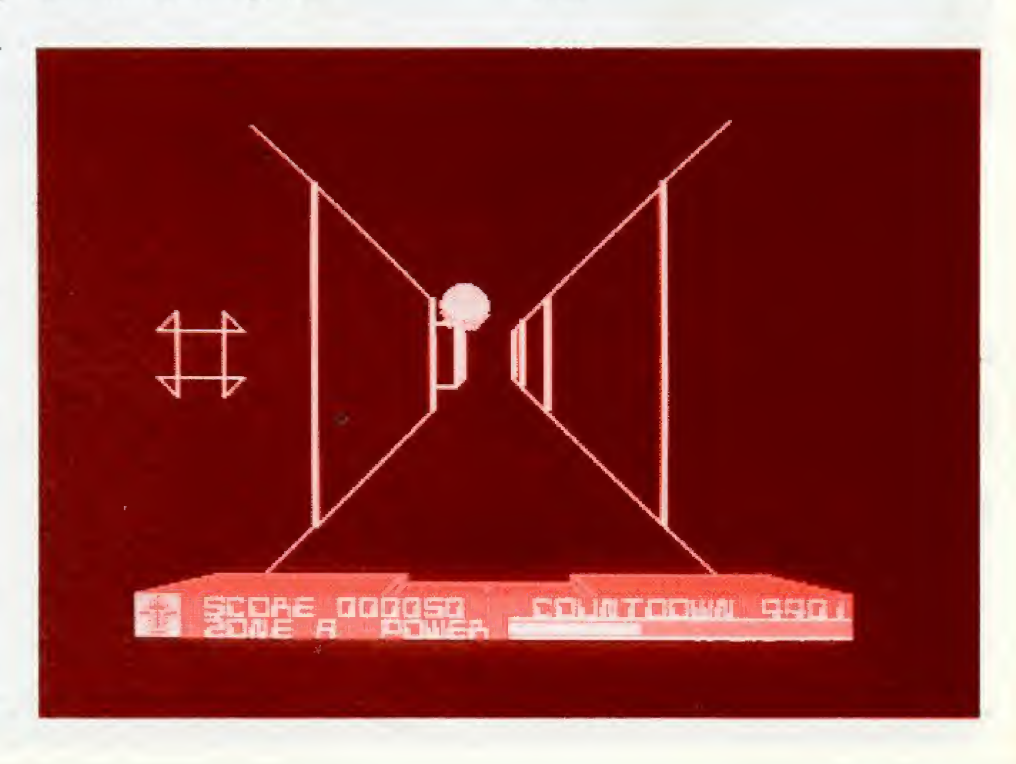




VECTRON 3D

FIREBIRD
SPECTRUM 48 K

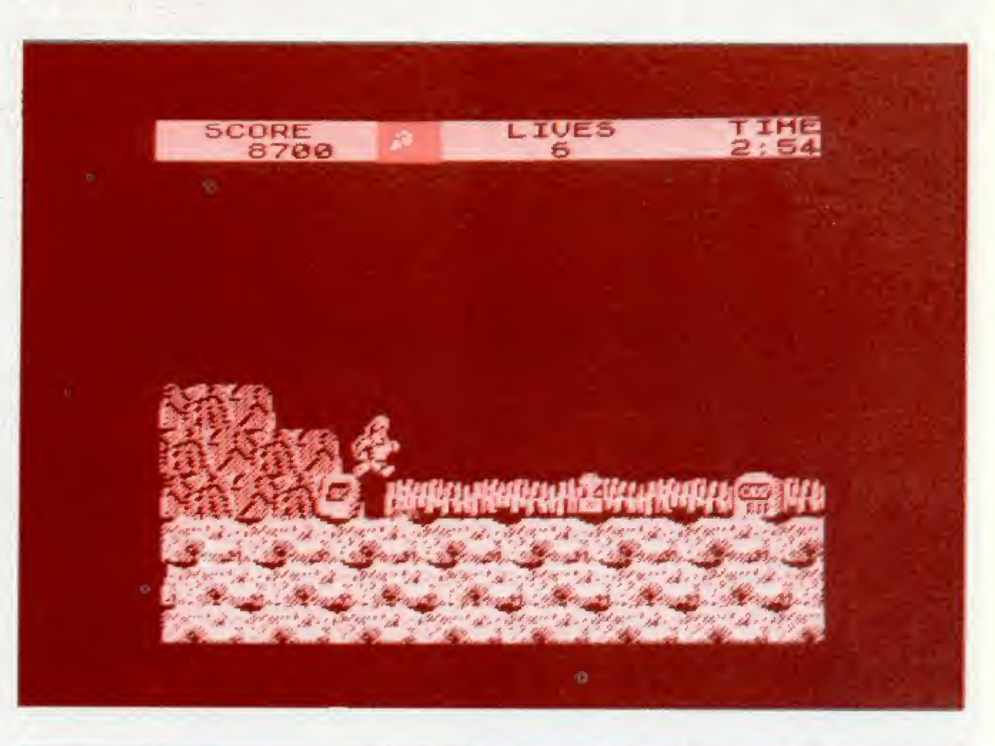
Nos encontramos con un 3D de lo más puro, uno de esos juegos en los que se sacrifica casi todo por conseguir una buena sensación de tridimensionalidad y perspectiva. Se trata de un clásico laberinto en el que sólo nos encontraremos con un sencillo punto de mira, las líneas que representan las paredes, y unos enemigos no tan esquemáticos como nos tienen acostumbrados en este tipo de juegos. La velocidad conseguida es muy grande para la relativa calidad de los gráficos y su interesante animación.
El conducir nuestros Spectrum por tan extraño laberinto en cualquier cosa menos fácil, aunque con unas buenas dosis de concentración podemos conseguir cierto control de vez en cuando. La sensación es buena, y la cosa se pone aún más interesante cuando conseguimos ponernos a la cola de algún enemigo para acribillarle con las nutridas ráfagas del láser de abordo.
De todas formas la velocidad de nuestra nave es excesiva para quien comienza (y también para el que no es tan novato), y desgraciadamente no puede ser modificada al gusto del jugador, que suele acabar chocando contra las paredes y las naves que circulan en sentido contrario a los pocos segundos. La opción «map», que sobreimpresiona un plano del laberinto y sus «habitantes» no sirve de mucho, pues la nave sigue moviéndose a su enorme velocidad, y no se puede prestar atención a las dos cosas.
Dejando a un lado esos pequeños grandes detalles que lo hacen tan complicado para nuestras humanas mentes, hay que hablar de un juego bastante logrado técnicamente y que, a la vez que puede catalogarse como un clásico, se sale bastante de la forma de ser de la mayoría de los títulos que están apareciendo últimamente. Un juego que rebosa acción por los cuatro costados y que pondrá las cosas difíciles a quienes se las den de imbatibles en programas de este tipo.

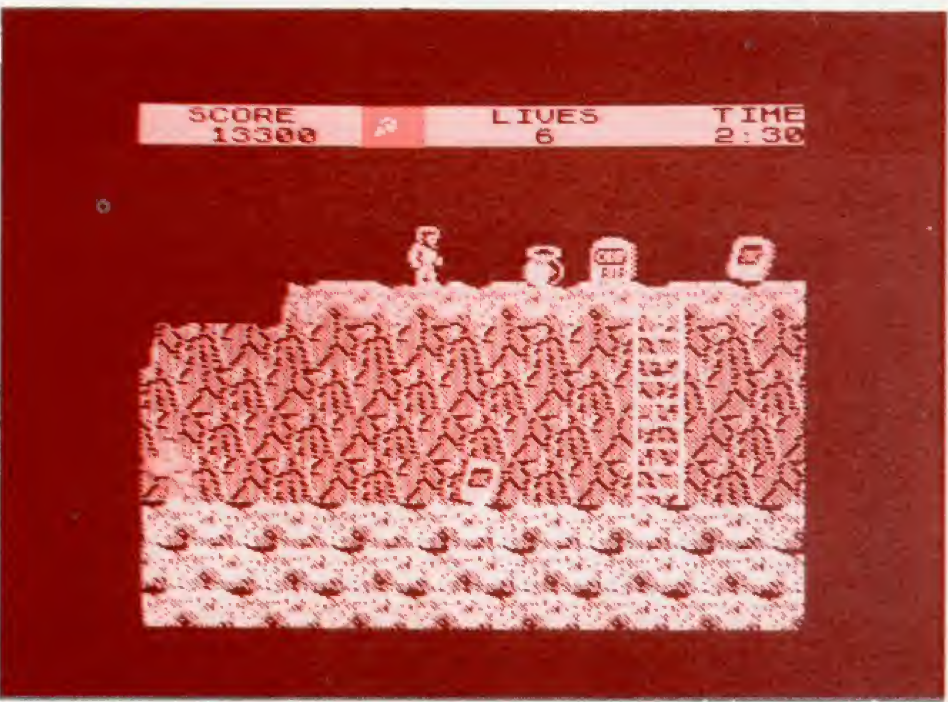


GHOST'N GOBLINS

ELITE

SPECTRUM 48 K





Por fin ha visto la luz la versión para Spectrum de este juego, tras haber pasado por las manos de millones de jugadores en las típicas «máquinas de marcianitos» de miles de bares y salas de recreativos. Se repite esa maniobra que ya nos resulta familiar como es lanzar versiones de estos juegos para todos los micros domésticos, justo cuando su nivel de atracción en ese otro mercado comienza a disminuir, y aprovechando que todavía son muchos los «viciosos» que darán cualquier cosa por tener su programa favorito en casa y no tener que soltar «cinco pavos» cada vez que deseen hechar una partida. Parece que Elite se está convirtiendo en una especialista en esto como pudo comprobarse no hace mucho con esos «bombazos» que fueron (y siguen siendo) Commando y Bomb Jack. Como todos los que tienen esa procedencia, se trata de un juego en el que prima la acción. Todo su atractivo reside en unos sprites «simpáticos» que se mueven muy rápida y muy suavemente. Un secreto que empieza a ser de dominio público y que no falla: denle al jugador un personaje sencillo pero con mucha velocidad de reacción, de forma que consiga «sentirse él», y le harán feliz.

Nuestro protagonista deberá moverse a lo largo de cementerios, riscos y otros lúgubres parajes en busca de la bella princesa que se halla presa de un pérfido y endemoniado señor feudal. Habrá de tener cuidado de esquivar o destruir a los abundantes «zombis» que surgen de las entrañas de la tierra con el objeto de acabar con las escasas vidas de que dispone nuestro héroe. Otros enemigos también se disputarán el acabar con nosotros, por lo que deberemos ser generosos en el uso de las inacabables armas de que disponemos.

El acabado general del programa no es del todo malo, pero ha sido bastante recortada la versión original para conseguir acoplarla al nuevo ordenador. Lo que más ha cambiado ha sido la parte gráfica, sobre todo lo referente a los colores, dadas las limitaciones del Spectrum en este sentido. Esto es un fuerte revés para este juego y desde luego rompe mucho el «encanto» que éste tenía para ese público que lo puso en la cúspide en su día. Todo un «matar o huir para sobrevivir», destroza los nervios de cualquiera a las pocas partidas. De eso se trata, ¿no?

Compresión de textos

A la hora de almacenar textos en la memoria del ordenador, lo más corriente es colocar sobre cada una de las posiciones de la RAM un código ASCII exclusivamente, sin más complicaciones. De esta forma, si 35 Kb corresponden a 35840 bytes, utilizando la misma cantidad de memoria se podrán almacenar 35840 caracteres codificados en ASCII. Pero, todo sea por aprovechar hasta el último byte de nuestro micro, esta situación puede cambiar ahorrando una tremenda cantidad de memoria gracias a una sencilla rutina en código máquina.



El juego de caracteres del SPECTRUM está formado por 96 letras, números y signos especiales. Cada uno de estos caracteres está definido por un código ASCII comprendido entre 32 y 127.

Si nos fijamos en el formato binario de esta codificación, veremos que el 7.º bit nunca es utilizado; el código más alto, 127, corresponde en binario a 01000000.

De momento esto supone ya un cierto derroche de memoria, aunque no en grandes cantidades, al estar cada dirección de la RAM compuesta por 8 bits.

Se podría hacer una rutina que aprovechara al máximo esos 8 bits (ya veremos más adelante cómo se hace) utilizando el juego entero de caracteres pero no merece la pena ya que sólo nos ahorraríamos un byte por cada 8. Es decir, que utilizando solamente 7 bits para cada carácter, lograríamos almacenar en 35 Kb de memoria 40960 caracteres. De todas maneras, supondría ya un ahorro de 5120 bytes, pero somos muy «agarrados» y queremos exprimir nuestro paciente SPECTRUM hasta límites insospechados.

Mutilación del juego de caracteres

Con nuestra manía de ahorrar, vamoa a ver ahora que tal nos iría empleando para cada carácter solamente 6 de los 8 bits que conforman su código.

Lo primero que deducimos consiste en que el juego de caracteres quedaría reducido a 64 combinaciones diferentes (entre 0 y 63 decimal). Tomemos el manual del ordenador y consultemos el apéndice A (tabla del juego de caracteres). Si contamos los 64 caracteres disponibles a partir del código 32 (espacio), veremos que podemos disponer de todos los números, letras mayúsculas y signos más importantes. Las minúsculas quedarían excluídas pero lo que nos queda es más que suficiente para emplearlo en un pequeño procesador de textos sin grandes



ambiciones estéticas. La imposibilidad de utilizar las letras minúsculas se compensa por el hecho de que nos vamos a ahorrar un byte por cada 4 caracteres. ¡Siguiendo con el mismo ejemplo, en 35 Kb de memoria podremos almacenar hasta 47786 caracteres! Conseguiremos eludir a la «Hacienda» de nuestro ordenador 16946 bytes netos.

Todo esto está muy bien, pensará el lector, ¿pero cómo es posible almacenar 4 caracteres en tres bytes? Esto es lo que veremos a continuación.

Un carácter y un tercio en un byte

Si el Z-80 tuviera una configuración de 6 bits no habría ningún problema para lo que queremos hacer (pero repercutiría gravemente a la hora de programar al ofrecer un juego de instrucciones menor). El asunto está en aprovechar los 8 bits de cada una de las localizaciones de memoria para que pueda caber un código (el cual estará formado por 6 bits) y una parte de otro. La figura 1 muestra cómo puede conseguirse este fin. De esta manera, por cada tres grupos de bytes en memoria, logramos empaquetar 4 caracteres. El primer byte del grupo contiene un carácter completo más un tercio del siguiente. El segundo, dos tercios del segundo carácter y dos tercios del tercero, y el tercer byte un tercio del tercer carácter más el cuarto carácter completo. Para entendernos con las explicaciones que vendrán seguidamente vamos a llamar a este grupo comprimido de caracteres «paquete».

Compensión/codificación

Habíamos quedado en que solamente utilizaríamos los caracteres cuyos códigos están comprendidos entre 32 y 95. Aparentemente esto supone un ligero problema, ya que

Compresión de textos

entonces deberíamos emplear 7 bits para cada carácter. El problema deja de existir si cambiamos la codificación. Al operar con sólo 64 caracteres, podríamos muy bien asignar el código 0 para el primer carácter de la lista y 63 para el último. En la práctica, la rutina resuelve esto restando 32 del código ASCII antes de proceder a su tratamiento.

La rutina codificadora utiliza un buffer cuya dirección está contenida en la etiqueta START (esta dirección la puede variar el usuario «pokeando» en las direcciones 65002 y siguiente). En este buffer es donde se va a almacenar cada una de las páginas de texto que queremos comprimir. Una vez terminada una página de texto (que consta de 672 bytes, es decir, una pantalla completa), se procede a su compresión. Realizada su tarea, la rutina codificará la página en 168 paquetes de 3 bytes (504 bytes) a partir de la dirección contenida en la etiqueta TEXTO (dirección 65000 y siguiente). En esta dirección, al utilizar por primera vez la rutina, el usuario deberá marcar en formato de dos bytes la dirección de inicio del texto comprimido. Cada vez que se vayan codificando

Por cada tres grupos de bytes en memoria, logramos «empaquetar» cuatro caracteres.

más páginas de texto, el contenido de START se irá incrementando en 504.

Rotaciones e instrucciones lógicas

La filosofía de la rutina se caracteriza por el empleo constante de las instrucciones lógicas (AND y OR) y las rotaciones. El programa toma el primer byte de la dirección cocntenida en la etiqueta START, le resta 32 a su contenido y realiza dos rotaciones a la izquierda (RLCA) de tal forma que el formato binario del código se sitúe en el extremo izquierdo del byte. Seguidamente se accede al contenido de la siguiente dirección, se le resta igualmente 32 y se procede a una operación AND para poner a 0 los bits 0, 1, 2 y 3 del byte en cuestión (AND 00110000) preservando el contenido de los bits 4 y 5. Inmediatamente se realizan cuatro rotaciones a la derecha (RRCA) para colocarlos en el extremo derecho. Después de recuperar el anterior byte tratado que había quedado a salvo en el Stack, una operación OR hace que se mezclen en un mismo byte aquellos 6 bits con los dos de ahora. El resto del programa se basa en acciones similares, hasta haber formado un paquete. Un bucle condicional hace que se repita la misma operación 168 veces (168x4 =672 caracteres) gracias al contador situado en la etiqueta CONTA. Una vez terminado todo el proceso, la etiqueta TEXTO contiene ya la siguiente dirección a partir de donde se almacenará la siguiente página comprimida.

Decodificación e impresión

La rutina anterior no serviría de nada si no se dispusiera de otra que decodificara los caracteres compri-

```
   1 REM ORLANDO ARAUJO MARTIN
   2 REM   PRUEBA COMPRESOR
   3 REM   22 JUNIO 1986
   4
   5 GO SUB 900
   9 LET START=29000
  12 LET TEXTO=30000
  19 POKE 65001,INT (TEXTO/256):
 POKE 65000,TEXTO-(INT (TEXTO/25
6)*256)
  20 POKE 65003,INT (START/256):
 POKE 65002,START-(INT (START/25
6)*256)
 400 POKE 23728,32: LET N=0
 500 RANDOMIZE USR 65189
 510 RANDOMIZE USR 65008
 525 GO SUB 720
 605 LET r=PEEK 23728: LET r=r+1
: POKE 23728,r
 610 IF PEEK 23728=96 THEN  GO T
O 700
 630 LET N=N+1: GO TO 500
 700 CLS : PRINT "QUE PAGINA DES
EA CONSULTAR?": INPUT ">";N
 710 IF N<0 OR N>63 THEN  GO TO
700
 711 GO SUB 720
 713 PAUSE 0: CLS : GO TO 700
 720 LET PAG=TEXTO+(N*504): POKE
 65007,INT (PAG/256): POKE 65006
,PAG-(INT (PAG/256)*256): RANDOM
IZE USR 65099
 725 RETURN
 900 LET con=0: FOR i=65000 TO 6
5187: READ a: LET con=con+a: POK
E i,a: NEXT i
 910 IF con<>23575 THEN  PRINT
FLASH 1;"ERROR EN DATAS RUTINA C
OMPRESORA": STOP
 920 REM  cargador codigos
 921 RESTORE 2000: LET con=0: FO
R i=65189 TO 65212: READ a: LET
con=con+a: POKE i,a
 922 NEXT i
 923 IF con<>2361 THEN  PRINT "e
rror en datas cargador codigos":
 STOP
 950 RETURN
1001 DATA 40,119,72,113,0,0,48,1
17,1,168,0,237,67,236,253,42,234
,253,237,91,232,253,126,214,32
1002 DATA 7,7,245,35,126,214,32,
230,48,15,15,15,15,193,176,18,19
,126,214,32,230,15,7,7,7
1003 DATA 7,245,35,126,214,32,23
0,60,15,15,193,176,18,19,126,214
,32,15,15,230,192,245,35,126,214
1004 DATA 32,193,176,18,19,35,23
7,75,236,253,11,237,67,236,253,1
21,184,32,184,237,83,232,253,201
,205
1005 DATA 107,13,62,2,205,1,22,4
2,238,253,17,168,0,126,245,230,2
52,15,15,198,32,229,213,215,209
1006 DATA 225,241,230,3,7,7,7,7,
245,35,126,230,240,15,15,15,15,1
93,176,198,32,229,213,215,209
1007 DATA 225,126,230,15,7,7,245
,35,126,230,192,7,7,193,176,198,
32,229,213,215,209,225,126,230,6
3
1008 DATA 198,32,229,213,215,209
,225,35,27,122,187,32,181
1009
2000 REM datos cargador codigos
2100 DATA 22,21,58,176,92,95,42,
234,253,6,32,115,35,16,252,6,32
2110 DATA 21,122,254,0,32,244,20
1
```

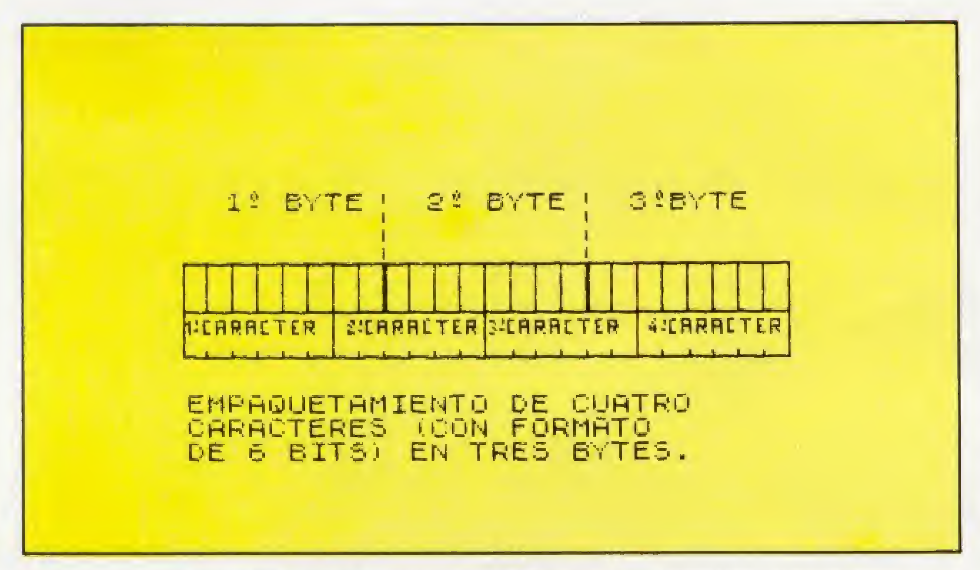
```
;ORLANDO ARAUJO MARTIN
;CODIFICADOR TEXTO
;17 de junio de 1986
;
;
TEXTO  EQU  65000
START  EQU  65002
CONTA  EQU  65004
PAG    EQU  65006

       ORG  65008
       ENT  $
       LD   BC,168
       LD   (CONTA),BC
;contador caracteres
       LD   HL,(START)
;direccion comienzo buffer
 texto a codificar
       LD   DE,(TEXTO)
;carga en DE la direccion
 actualizada texto a codificar
BUCLE  LD   A,(HL)
       SUB  32
;carga en A el codigo del
primer caracter del grupo
       RLCA
       RLCA
;2 desplazamientos a la iz
quierda del caracter
       PUSH AF
       INC  HL
;acceso al siguiente carac
ter del buffer
       LD   A,(HL)
       SUB  32
;carga del caracter
       AND  %00110000
;se toman los dos ultimos
bits del caracter
       RRCA
       RRCA
       RRCA
       RRCA
;y se desplazan hacia la d
erecha
       POP  BC
;recuperacion del primer c
racter
       OR   B
;mezcla de los 6 bits del
primer caracter con los
;dos primeros del segundo
       LD   (DE),A
;y se almacenan al estar y
a codificados
       INC  DE
;acceso a la siguiente dir
eccion del texto codificado
       LD   A,(HL)
       SUB  32
;se vuelve a tomar el segu
ndo caracter
       AND  %00001111
;considerando solamente lo
s cuatro primeros bits
       RLCA
       RLCA
       RLCA
       RLCA
;para rotarlos a la izquie
rda
       PUSH AF
       INC  HL
       LD   A,(HL)
       SUB  32
;acceso y carga al tercer
caracter
       AND  %00111100
;tomando en consideracion
 los cuatro ultimos bits
       RRCA
       RRCA
;2 rotaciones a la derecha
       POP  BC
       OR   B
;recuperacion de los 4 pri
meros bits del segundo ca-
;racter y mezcla con los 4
 ultimos del tercero
       LD   (DE),A
;y almacenamiento del conj
unto
       INC  DE
       LD   A,(HL)
       SUB  32
;acceso de nuevo al tercer
 caracter
       RRCA
       RRCA
;2 rotaciones a la derecha
 del caracter
       AND  %11000000
;se salvan los dos primero
s bits del caracter
       PUSH AF
       INC  HL
       LD   A,(HL)
       SUB  32
;acceso al cuarto caracter
       POP  BC
       OR   B
;y mezcla con los dos prim
eros bits del anterior
       LD   (DE),A
       INC  DE
;almacenamiento del tercer
 codigo del grupo
       INC  HL
       LD   BC,(CONTA)
       DEC  BC
       LD   (CONTA),BC
       LD   A,C
       CP   B
       JR   NZ,BUCLE
       LD   (TEXTO),DE
       RET

;RUTINA DECODIFICADORA

       CALL #0D6B
;borra pantalla
       LD   A,2
       CALL #1601
;abre canal 2
       LD   HL,(PAG)
;pagina elegida
       LD   DE,168
;contador caracteres
LOOP   LD   A,(HL)
       PUSH AF
       AND  %11111100
       RRCA
       RRCA
       ADD  A,32
       PUSH HL
       PUSH DE
       RST  #10
       POP  DE
       POP  HL
       POP  AF
       AND  %00000011
       RLCA
       RLCA
       RLCA
       RLCA
       PUSH AF
       INC  HL
       LD   A,(HL)
       AND  %11110000
       RRCA
       RRCA
       RRCA
       RRCA
       POP  BC
       OR   B
       ADD  A,32
       PUSH HL
       PUSH DE
       RST  #10
       POP  DE
       POP  HL
       LD   A,(HL)
       AND  %00001111
       RLCA
       RLCA
       PUSH AF
       INC  HL
       LD   A,(HL)
       AND  %11000000
       RLCA
       RLCA
       POP  BC
       OR   B
       ADD  A,32
       PUSH HL
       PUSH DE
       RST  #10
       POP  DE
       POP  HL
       LD   A,(HL)
       AND  %00111111
       ADD  A,32
       PUSH HL
       PUSH DE
       RST  #10
       POP  DE
       POP  HL
       INC  HL
       DEC  DE
       LD   A,D
       CP   E
       JR   NZ,LOOP
       RET
```

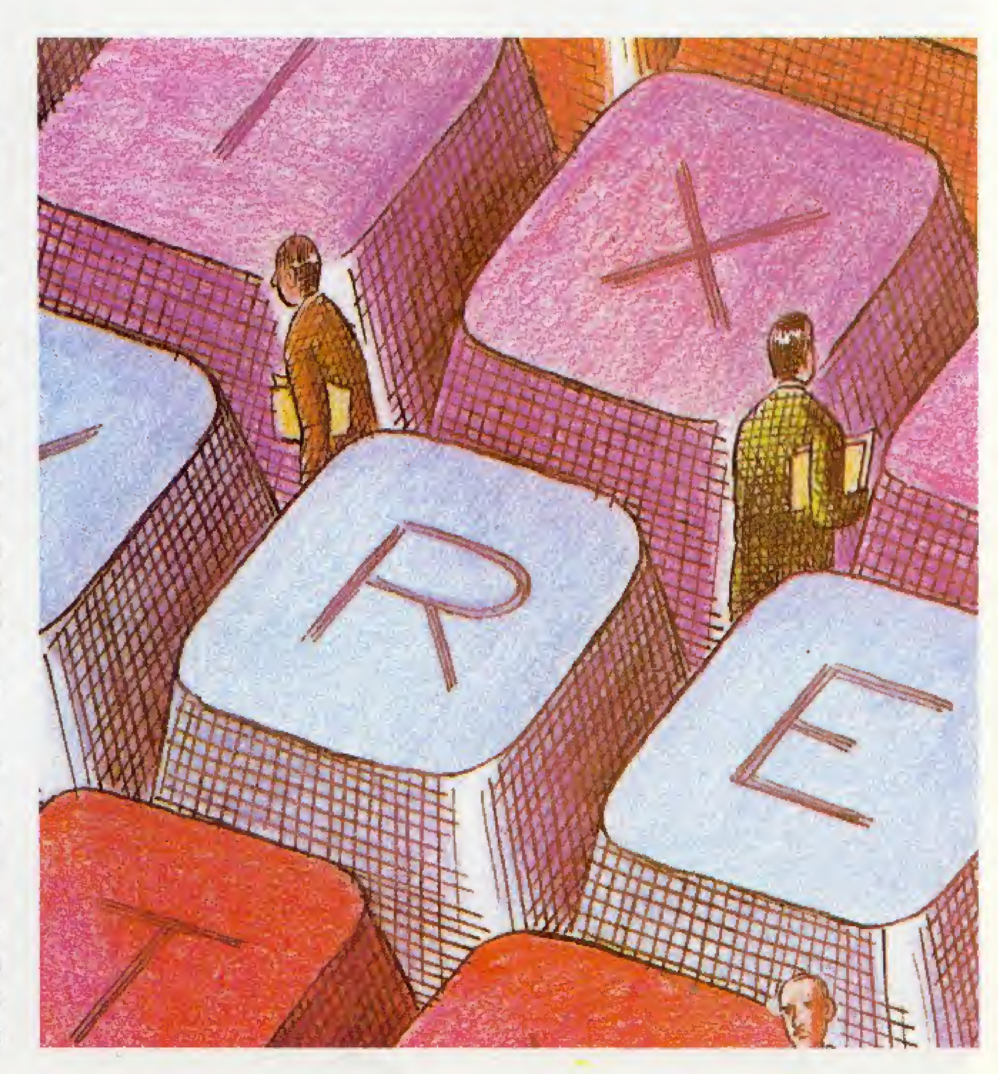
Compresión de textos



midos y los imprimiera. Gracias a ella podremos acceder a una página cualquiera del texto y ver su contenido.

Como cualquier rutina de impresión, ésta comienza llamando a la rutina del sistema operativo OD6BH que limpia la pantalla y se abre seguidamente el canal 2 de impresión. En el registro HL se almacena la página que deseamos consultar, cuyo dato está contenido en la etiqueta declarada al comienzo del programa; PAG. En esta dirección (65006), antes de llamar a la rutina decodificadora, el usuario debe almacenar en formato de dos bytes la dirección de inicio de la página a consultar. Por supuesto, para hacer las cosas más fáciles, en el programa BASIC que controle la rutina, sólo deberá introducirse el número de página mediante un INPUT (siendo 0 la primera página). Un sencillo cálculo en una línea BASIC se encargará de calcular la dirección a la que corresponde la página según el dato introducido y POKEará los datos resultantes a partir de la etiqueta PAG.

En DE colocamos el contador que inicializamos a 168 (168x3=504 bytes a decodificar). Se toma el primer byte de la página eliminándose el contenido de los dos primeros bits (AND 11111100). Se rota el byte dos posiciones a la derecha y se le suma 32 a su contenido. De esta manera conseguimos obtener el código ASCII original. Ahora ya no tenemos mas que imprimirlo aprovechando la llamada a la ROM RST 10H. ¡Pero cuidado! Siempre, antes de llamar una rutina ROM, debemos preservar el contenido importante de los registros que estemos utilizando en la rutina. En este caso, debemos de poner a salvo HL (puntero del texto comprimido) y DE (contador del bucle) y llamar inmediatamente a la rutina de impresión que imprimirá en la pantalla el código contenido en A. Entonces ya podemos volver a recuperarlos del Stack mediante POP. El resto del programa es semejante al anterior por lo cual no nos alargaremos demasiado. Al igual que aquél, se hace uso de las instrucciones de rotación y las operaciones lógicas. Una vez realizada una pasada dentro del bucle, se habrán imprimido 4 caracteres que anteriormente estaban firmemente compactados en un paquete de tres bytes.

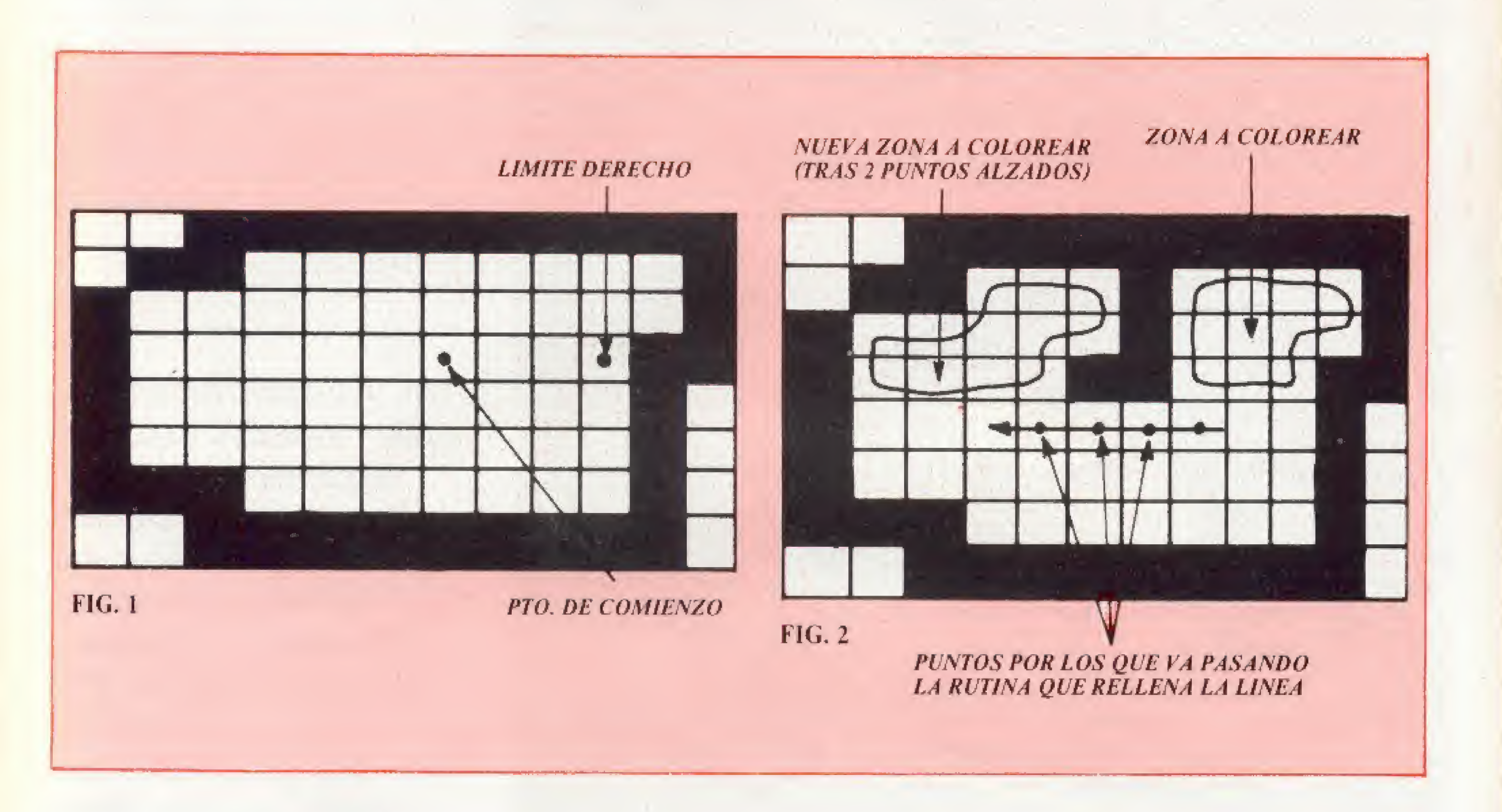




FILL ILUSTRADO

Las presentes rutinas en código máquina han sido diseñadas para poder rellenar (Fill) cualquier figura con posibilidad de utilizar entramados definidos por nosotros. Ocupan un total de 415 bytes, no son reubicables y han sido situadas juntas a partir de la dirección 64900 (para dejar sitio a los UDGs). Estas rutinas son la coloreadora propiamente dicha y la rutina entramadora.

FILL ILUSTRADO



La rutina coloreadora se utiliza definiendo una función de la forma:

DEF FN n(a,b,c)=USR 64900

y llamándola con una orden como:

LET a=FN n(X,Y,cc)

siendo *n* el nombre de la función, *X* e *Y* las coordenadas de un punto cualquiera en el interior de la figura a colorear, y *cc* una condición. Si *cc* vale 1, entonces además de colorear se crea una tabla especial e imprescindible si se desea luego utilizar algún tipo de entramado. Si *cc* vale 0 no se crea esta tabla ahorrando memoria (la tabla ocupa una gran cantidad de memoria) y permitiendo al uso de la rutina en programas largos o bien en máquinas de 16K —con el previo ajuste de la rutina y reubicación de la misma.

Esta tabla se crea desde el comienzo de la rutina (dirección 64900) hacia abajo. En caso de que la tabla bajara más allá del RAMTOP se provocaría el error "out of memory". Por tanto debemos previamente introducir una orden CLEAR nn, dejando el espacio necesario a la tabla. En caso de usar la opción de no creación de la tabla (cc=0) no debemos preocuparnos de esto último.

La tabla utilizada para el entramado ocupa mucha memoria, por lo que en Spectrum 16 K debe suprimirse.

Además, la rutina devuelve al BASIC en el registro doble BC el número de puntos que han sido PLOTeados. Así, si ejecutamos la orden:

LET a=FN n(X,Y,cc)

en la variable *a* tendremos el número de los puntos que «caben» dentro de la figura.

Esto puede ser útil —a parte de la mera curiosidad— para calcular, por ejemplo, la superficie de cualquier «cosa» por complicada que sea con sólo dibujarla en la pantalla y colorearla; en ese caso la superficie vendría dada en «pixels cuadrados» que en función de la escala se podrían pasar a las unidades que fueran necesarias.

Otra curiosidad es que el rellenado de una figura puede ser parado en cualquier momento pulsando la tecla «SPACE». Esta parada será siempre tras el relleno de una línea completa de la figura. El resto de las funciones, como la posibilidad de uso de entramados o la cuenta del número de puntos rellenados, funcionarán igual que si la figura se terminara allí.

La rutina entramadora se utiliza también definiendo una función de la forma:

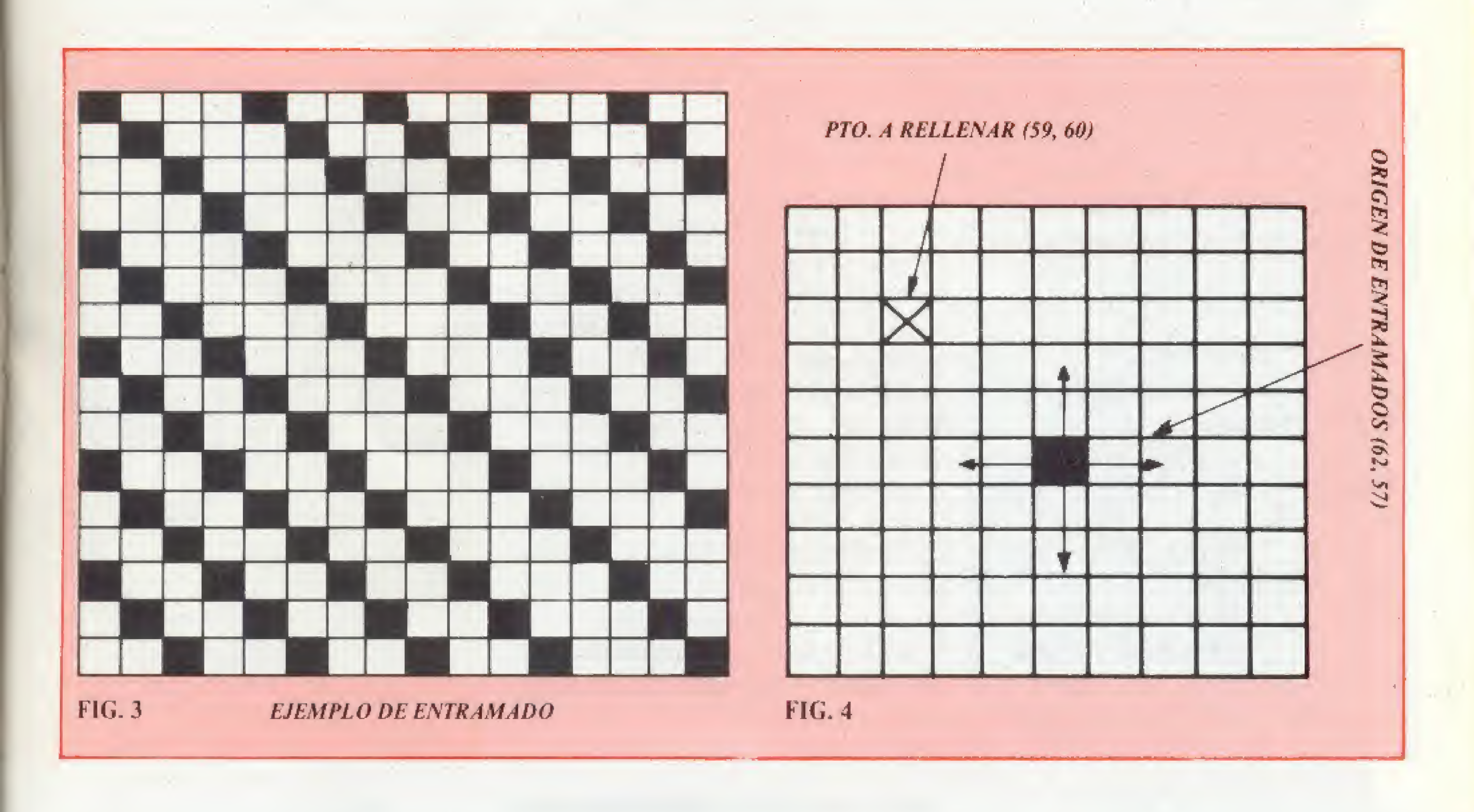


FIG. 3 *EJEMPLO DE ENTRAMADO*

FIG. 4

DEF FN n(a)=USR 64914

y llamándola con una orden como:

LET a=FN n(dir))

siendo *dir* la dirección donde se halla situado el entramado dentro de la memoria. El entramado es un gráfico de 16 x 16 pixels que por tanto ocupa un total de 32 bytes. Este gráfico se almacena en memoria por filas de 16 bits, siendo la primera fila la que se define de izquierda a derecha por los dos primeros bytes; los dos siguientes definirían la segunda, los dos siguientes la tercera y así sucesivamente.

Como mucha gente no está habituada a esta forma de almacenar los gráficos, se dispone también de la posibilidad de utilizar los UDGs A, B, C y D para definir el entramado. En este caso la figura estaría formada por estos cuatro gráficos siendo la mitad superior la formada por los gráficos A y B y la mitad inferior la formada por los gráficos C y D. Si se

El entramado puede definirse usando los UDG A, B, C y D. Si se elige esta opción, la rutina lo adapta al otro formato.

elige esta opción, la dirección que se debe indicar como inicio del gráfico es la 0, encargándose la rutina de adaptar el gráfico al otro formato depositándolo en la memoria intermedia de la impresora. Así, si se desea adaptar un gráfico del formato UDGs al otro bastará con entramar una figura pequeña o bien introducir el siguiente POKE:

POKE 64899,255

de esta forma no se entramará nada, realizándose sólo el proceso de adaptar el gráfico. Una vez hecho esto (con el parámetro *dir*=0) bastará con trasladar esos 32 bytes que forman la figura a donde convenga.

A diferencia de la otra rutina, la entramadora no retorna en BC el número de puntos, por lo que el valor de la función no tendrá ningún sentido. Añadir además que se puede cambiar el entramado cuantas veces se quiera mientras no se destruya la tabla coloreando una nueva figura con la opción *cc*=1.

Funcionamiento

El funcionamiento de las rutinas es algo complejo por lo que aconsejo, a los que quieran «atreverse» con el listado ensamblador, que primero lean estas líneas.

A) El funcionamiento de la rutina coloreadora es simple en teoría, aunque se complica un tanto en el listado ensamblador.

Para colorear la figura se utiliza un STACK que sirve para guardar las coordenadas de las zonas o líneas que se dejan para «más ade-

FILL ILUSTRADO

TABLA DE POKES:
(Para ahorrar entramados)

Figura Normal
POKE 65220,31
Figura Horizontal
POKE 65220,23
Figura Normal POKE
65263,122
POKE 65264,144
Espejo Vertical
POKE 65263,120
POKE 65264,146

lante» mientras se colorea otra zona distinta. Como STACK se usa el del Z-80 con órdenes tan simples como PUSH o POP.

Para seguir el proceso hay que tener presente las figuras.

1. Se comprueba si el punto en las coordenadas actuales está alzado (POINT (x,y)=1). En caso afirmativo la zona en cuestión no tendría nada que colorear, por lo que se pasaría a la siguiente zona.

2. Se busca el límite derecho de la línea a colorear (ver fig. 1).

3. A continuación se introduce en la tabla (si la condición *c* vale 1) la coordenada Y y la coordenada X final (del límite derecho).

Esta tabla está formada por grupos de 3 bytes, siendo el primero la coordenada Y y el segundo y tercero las coordenadas X_{final} y $X_{inicial}$ respectivamente. Caso de que la coordenada Y sea 255 (imposible) se toma como final de la tabla por la rutina entramadora, siendo esta la señal.

Esta tabla se completa introduciendo el tercer byte cuando se termina el rellenado de una línea.

4. A continuación se inicia el rellenado de la línea (de derecha a izquierda). Primero se PLOTea en las coordenadas actuales y luego se comprueban los puntos inmediatamente inferior y superior. Si no hay nada (POINT (x,y)=0), entonces se almacenan en el STACK las coordenadas de ese punto (vacío) y se activa o alza un banderíz. Este banderín evita que el STACK se llene de coordenadas innecesarias para el completo rellenado de la figura. Previamente a la introducción de las coordenadas en el STACK se comprueba este banderín y si está alzado no se realiza la introducción.

En caso de que el punto esté alzado, entonces se baja dicho banderín porque esto quiere decir que puede venir a continuación una nueva zona (ver fig. 2). Esta comprobación se realiza tanto en la zona superior como en la inferior, pues puede suceder en ambas lo mismo.

La rutina entramadora se limita a leer la tabla creada por la otra rutina y poner o quitar los puntos correspondientes.

Cuando se termina de rellenar una línea repitiendo el mismo proceso y decrementando la coordenada correspondiente, se pasa a una nueva zona. (Una línea se completa al encontrar un punto alzado o bien el borde de la pantalla).

Para pasar a una nueva zona o línea, se ponen a cero los banderines y se sacan del STACK las nuevas coordenadas. Hecho esto se comprueba si estas coordenadas son la señal previamente introducida antes de comenzar a rellenar la figura. Si lo son, se retorna al BASIC devolviendo el número de puntos



PLOTeados; si no, se vuelve a iniciar el proceso de rellenado de una línea. Fíjese el lector que cuando se comprueba durante el rellenado de una línea si los puntos superior e inferior están a 0, si es necesario se introducen las coordenadas de dicho punto en el STACK; por lo que al



sacar las nuevas coordenadas del STACK se continúa en la línea que previamente había sido detectada como vacía y n es necesario que la rutina se encargue de subir o bajar una línea para seguir rellenando.

Como curiosidad, puede decir que si la rutina colorea primero de arriba a abajo, es porque en el STACK se introducen primero las coordenadas de los «huecos» que se detectan debajo, y luego los de arriba. Por ello, como el STACK es de tipo LIFO (last in, first out), las últimas coordenadas son las superiores.

B) El funcionamiento de la rutina entramadora se limita a leer la tabla creada por la otra rutina y poner o quitar los puntos que corresponda, siendo esto lo más complicado, por lo que paso a explicar cómo se sabe si se debe poner o quitar un punto.

Para empezar, se toma como origen de entramados las coordenadas del punto desde el que se comenzó a colorear la figura. Las fórmulas que se usan para saber qué punto se debe comprobar en el entramado para luego ser copiado en pantalla son las siguientes:

$$X = (CX_{origen} - CX) \text{ MOD } 16$$
$$Y = (CY_{origen} - CY) \text{ MOD } 16$$

$CX_{origen}-CX$ o $CY_{origen}-CY$ pueden ser negativos, pero como en código máquina los números negativos se expresan como complemento a dos, ni se tiene en cuenta. Veamos un ejemplo:

Tenemos un entramado cualquiera (fig. 3) y queremos saber qué le corresponde a un cierto punto

FILL ILUSTRADO

```
10              ORG   64000
20  FILL        LD    IX,(23651)
30              LD    B,(IX+3)
40              LD    C,(IX+2)
50  FILL2       PUSH  BC
60              LD    A,2
70              CALL  #1601
80              LD    IX,FLAGS
90              LD    A,1
100             LD    (INCY),A
110             LD    (INCX),A
120             LD    BC,65367
130             LD    (STACK),BC
140             LD    BC,65535
150             CALL  PUSH
160             POP   BC
170 F_BUC1      LD    (CC),BC
180             RES   0,(IX+0)
190             CALL  PLOT
200 F_BUC2      RES   1,(IX+0)
210 F_BUC3      LD    A,(INCY)
220             ADD   A,C
230             LD    C,A
240             CALL  POINT
250             JR    Z,F_PTO1
260             BIT   0,(IX+0)
270             JR    Z,F_PTO2
280             LD    A,(INCX)
290             ADD   A,B
300             LD    B,A
310             INC   C
320             LD    (CC2),BC
330             LD    BC,(CC1)
340             RES   3,(IX+0)
350             LD    A,(INCX)
360             ADD   A,B
370             LD    B,A
380 F_BUC4      CALL  POINT
390             JR    Z,F_PTO3
400             RES   3,(IX+0)
410 F_BUC5      LD    HL,(CC2)
420             AND   A
430             SBC   HL,BC
440             LD    A,H
450             OR    L
460             JR    Z,POP
470             LD    A,(INCY)
480             ADD   A,C
490             LD    C,A
500             JR    F_BUC4
510 F_PTO1      CALL  PLOT
520             PUSH  BC
530             LD    A,(INCX)
540             LD    D,A
550             LD    A,B
560             SUB   D
570             LD    B,A
580             CALL  POINT
590             POP   BC
600             JR    NZ,F_BUC2
610             BIT   1,(IX+0)
620             JR    NZ,F_BUC3
630             SET   1,(IX+0)
640             CALL  PUSH
650             JR    F_BUC3
660 F_PTO2      SET   0,(IX+0)
670             DEC   C
680             LD    A,255
690             LD    (INCY),A
700             LD    (CC1),BC
710             LD    BC,(CC)
720             JR    F_BUC2
730 F_PTO3      BIT   3,(IX+0)
740             JR    NZ,F_BUC5
750             SET   3,(IX+0)
760             CALL  PUSH
770             JR    F_BUC5
780 PUSH        LD    HL,(STACK)
790             LD    DE,ZFIN
800             INC   DE
810             PUSH  HL
820             AND   A
830             SBC   HL,DE
840             POP   HL
850             JR    NZ,F_SIG3
860 F_ERR       RST   8
870             DEFB  3
880 F_SIG3      LD    (HL),B
890             DEC   HL
900             LD    (HL),C
910             DEC   HL
920             LD    (STACK),HL
930             RET
940 POP         LD    HL,(STACK)
950             INC   HL
960             LD    C,(HL)
970             INC   HL
980             LD    B,(HL)
990             LD    (STACK),HL
1000            LD    A,B
1010            CP    255
1020            JR    NZ,F_SIG5
1030            LD    A,C
1040            CP    255
1050            JR    NZ,F_SIG5
1060            RET
1070 F_SIG5     LD    A,1
1080            LD    (INCY),A
1090            PUSH  BC
1100            LD    A,(INCX)
1110            LD    D,A
1120            LD    A,B
1130            SUB   D
1140            LD    B,A
1150            CALL  POINT
1160            POP   BC
1170            JP    NZ,F_BUC1
1180            LD    A,(INCX)
1190            LD    D,A
1200            XOR   A
1210            SUB   D
1220            LD    (INCX),A
1230            JP    F_BUC1
1240 POINT      LD    A,#7F
1250            IN    A,(#FE)
1260            AND   1
1270            JR    NZ,P_SIG2
1280            RST   8
1290            DEFB  12
1300 P_SIG2     PUSH  BC
1310            CALL  #22AA
1320            LD    B,A
1330            INC   B
1340            LD    A,(HL)
1350 P_BUC1     RLC   A
1360            DJNZ  P_BUC1
1370            AND   1
1380            NOP
1390            NOP
1400            POP   BC
1410            RET
1420 PLOT       PUSH  BC
1430            CALL  #22E5
1440            POP   BC
1450            RET
1460 FLAGS      DEFB  0
1470 INCX       DEFB  0
1480 INCY       DEFB  0
1490 CC         DEFW  0
1500 CC1        DEFW  0
1510 CC2        DEFW  0
1520 STACK      DEFW  0
1530 ZFIN       END
```

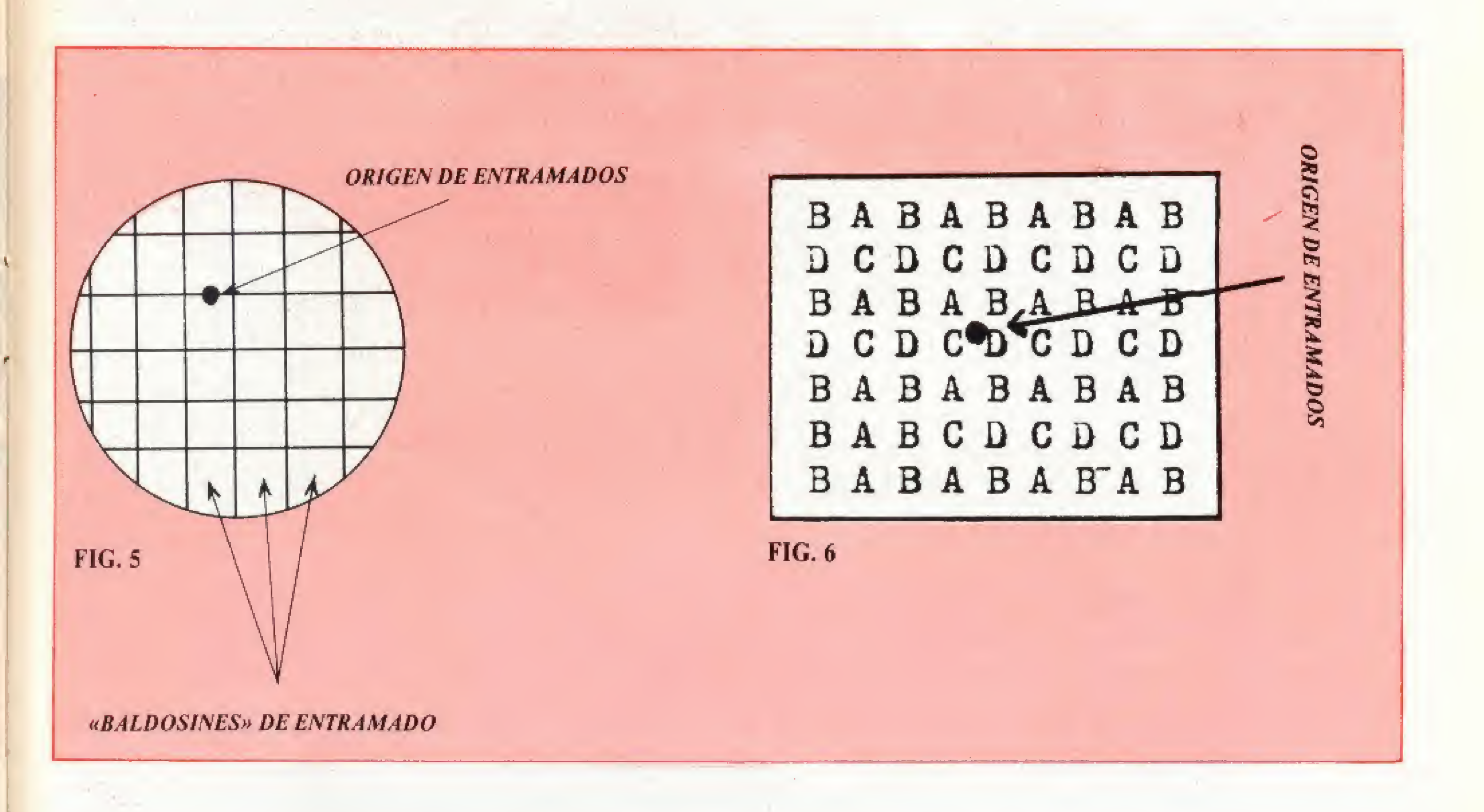


FIG. 5

FIG. 6

(59, 60) si el origen de entramados está en (62, 57). Para ello acudimos a las fórmulas:

X = (62 – 59) MOD 16
Y = (57 – 60) MOD 16

siendo X=3 e Y=13 (si no le da esto mire a ver si ha efectuado bien el complemento a dos para números negativos). Basta ahora con fijarnos en el estado del punto con coordenadas (3, 13) en el entramado, y sabremos que le corresponde un 1. (La X es la coordenada horizontal y la Y es la vertical).

Esto es algo así como si consideráramos el entramado como si fuera un baldosín y llenáramos con esos baldosines la figura. El origen de entramados es el punto de partida de los baldosines. Este origen coincide con el punto medio superior del baldosín (fig. 5) que está debajo del origen.

Si utilizáramos los UDGs A, B, C

Dado que la rutina entramadora lee de la tabla y no la destruye, se puede cambiar el entramado cuantas veces se necesite.

y D como entramado sin cambiarlos, veríamos algo así como lo que se aprecia en la figura 6 respecto al origen de entramados.

Dado que la rutina entramadora lee de la tabla y no la destruye, podemos cambiar el entramado cuantas veces queramos hasta que se cambie la tabla para otra figura. (O se anule ésta con el POKE anteriormente mencionado).

Como ejemplo de lo dicho prueba lo siguiente:

```
CIRCLE 127, 87, 85:
LET a=FN a (127, 87, 1)
```

(suponiendo que la función definida se llame *a*). Pruebe ahora:

```
LET a=FN b(dir)
```

(suponiendo que la función se llame *b*); y procure que *dir* tenga un valor dentro de la ROM. Podrá repetir esta orden cambiando *dir* cuantas veces quiera.

Pruebe ahora *dir*=15000. Podrá ver que rellena por completo, pues la dirección 15000 y siguientes están dentro de «agujero negro» de la ROM.

A los programadores que usen muchos entramados les aconsejamos que consultaran la tabla de POKES, que los guardaran directamente en memoria y que los salvaran al cassette en forma de BYTES en lugar de crearlos en el propio programa.

Fco. Javier Elices Vallejo

ROTACION DE UDGs

El mes pasado vimos cómo podía realizarse la inversión de gráficos definidos por el usuario en dos sentidos diferentes (horizontalmente y verticalmente). En el presente artículo trataremos un tema similar aunque algo más largo de llevar a buen término que el anterior. Con la rutina que vamos a desarrollar podremos orientar cualquier UDG en cuatro direcciones diferentes, lo cual supone un ahorro mayor que en el caso de la inversión.

Si el mes pasado lográbamos obtener 2 UDGs a partir de uno, con la actual rutina obtendremos 4 diferentes.

Este programa en Código Máquina permite orientar un UDG en 4 direcciones distintas. El efecto conseguido al llamar reiteradamente la rutina es una rotación del UDG, rotación que puede conseguirse a voluntad tanto de derecha a izquierda como de izquierda a derecha.

Una de las muchas utilidades que pueden deducirse a primera vista podría ser la simulación del movimiento rotatorio de las ruedas de un automóvil. En lugar de crear cuatro gráficos diferentes, bastaría con crear uno sólo y llamar a la rutina. Así de sencillo. De igual forma, si tiene pensado programar un «Comecocos», no tiene más que limitarse a crear una única figura del famoso «comefantasmas». Las otras tres orientaciones del «pac-man» las irá calculando la rutina por usted.

Existen otras muchas aplicaciones destinadas a juegos que con seguridad irá descubriendo según sus necesidades de programación.

La rutina utiliza cuatro variables diferentes que se almacenan en otras tantas direcciones de memoria

Ideas básicas de la rotación

De la misma forma que comenzamos en el anterior artículo, antes de nada debemos imaginarnos el proceso dejando a un lado el ordenador y los fríos mnemónicos Assembler, ya que si no se tienen las ideas bien claras, llegar a la solución del problema, programando directamente, puede ser un verdadero rompecabezas.

De inmediato nos encontramos con un problema que ofrece dos situaciones diferentes. Por un lado, podemos rotar el UDG de derecha a izquierda, y por el otro, de izquierda a derecha. Estudiemos de momento la primera situación.

Imaginemos un cuadrado dibujado en un papel dividido en 8x8 celdillas. Cada una de estas celdillas corresponden a los bits individuales de cada una de las filas o bytes del UDG. Numeramos secuencialmente cada una de las celdas del 1 al 64. Seguidamente, damos un cuarto de vuelta al papel (de derecha a izquierda). El siguiente paso consiste en estudiar los cambios que han sufrido los bits del UDG para obtener una regla común que será la que nos permitirá llegar al futuro algoritmo de rotación.



ROTACION DE UDGs

Esto es lo que descubrimos. El último bit de la primera fila ocupa la 1.ª posición de la primera fila; el último bit de la segunda fila ocupa el segundo lugar de la primera fila; el último de la tercera, el tercero de la primera, y así sucesivamente hasta haber completado la primera fila. Entonces, los siguientes bits que se trasladan son los penúltimos de cada una de las 8 filas que se van desplazando a los 8 lugares correspondientes de la segunda fila. El proceso se repite hasta haber completado la totalidad de las filas.

En el caso de que la rotación fuera de izquierda a derecha, el proceso sería inverso; el 1.º bit de la 8.ª fila se traslada al lugar que ocupaba el 1.º de la 1.ª fila; el último de la 7.ª fila va al lugar del segundo puesto de la primera, etc.

Esta es la base teórica de nuestra rutina. Teniendo presente estas ideas, ya podemos pasar a realizar el programa fuente sin grandes dificultades.

El programa fuente

La rutina hace uso de cuatro variables almacenadas a partir de otras tantas direcciones de memoria. En la dirección 59996, la rutina almacena un número que según sea 0 ó 1, accedemos a la rotación derecha-izquierda o izquierda-derecha respectivamente. En las 59999, introducimos el n.º de orden del UDG a rotar (0 a 20). La dirección 59997 y siguiente, la utiliza el propio programa para almacenar el comienzo de la dirección de una tabla a partir de la cual almacenamos una copia de los datos del UDG. Esta última dirección será utilizada por el registro índice IX.

La rutina puede llamarse de dos formas diferentes. Si deseamos una rotación derecha-izquierda, habrá que ejecutarla a partir de la dirección 60000, almacenándose en la dirección 59996 un 0 para saltar inmediatamente a la etiqueta START donde se procede a la multiplicación del n.º del UDG que ya vimos en el anterior artículo. Si la ejecución es a partir de la dirección 60007, se carga en la mencionada dirección un 1 y el programa sigue su curso a partir de START. Después de haber procedido a la ya comentada multiplicación por 8 (operación que sirve para tener un acceso inmediato al UDG elegido), un bucle LDIR carga a partir de la tabla salva una copia del UDG para tener un modelo intacto con el cual puedan efectuarse las comparaciones. Para tener acceso a estos datos, vamos a emplear el registro IX como puntero, con lo cual lo cargamos con la dirección original contenido en salva (líneas 30 a 32).



Seguidamente, cargamos de nuevo HL con el comienzo de la dirección del UDG e iniciamos una serie de contadores que nos servirán para llevar a cabo los constantes bucles dentro de los cuales se desarrollan las comparaciones de cada uno de los bits.

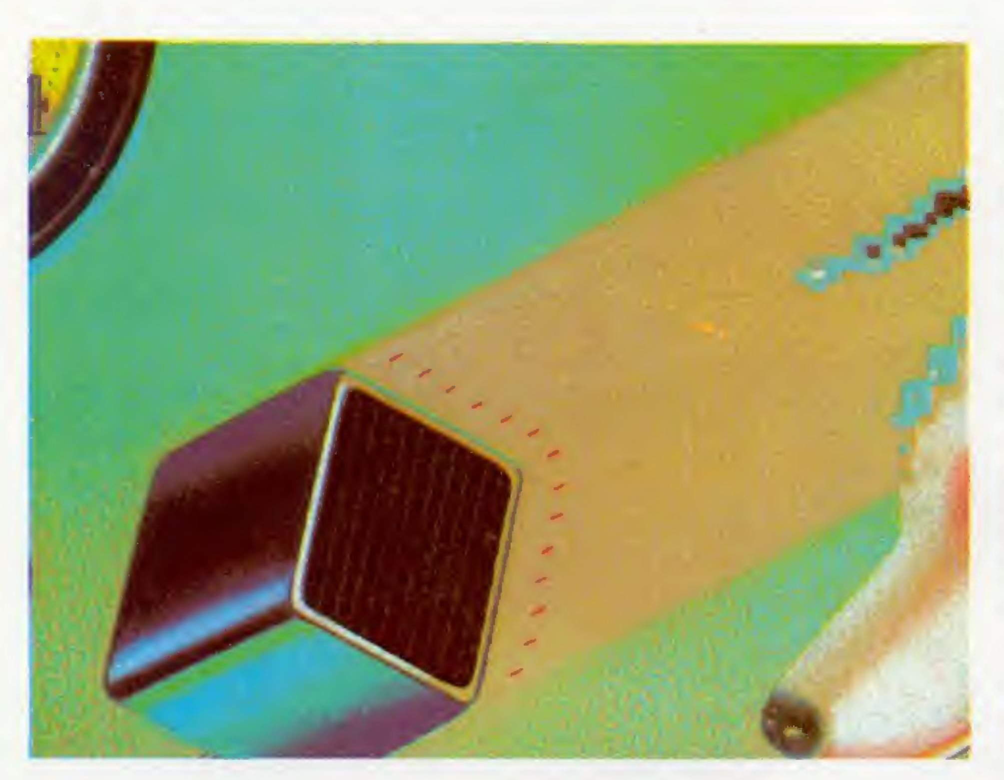
ROTACION DE UDGs

```
;Orlando Araujo
;ROTACION UDGs
;3 MAYO 1986


        ORG   60000
        ENT   $
        LD    A,0
        LD    (59996),A
        JR    START
;Rotacion derecha-izquierda
        LD    A,1
        LD    (59996),A
;Rotacion izquierda-derecha
START   LD    A,(59999)
        SLA   A
        SLA   A
        SLA   A
        LD    E,A
        LD    D,0
        LD    HL,(23675)
        ADD   HL,DE
;Acceso al UDG elegido mediante multiplicacion por 8 de n. UDG
        LD    (59997),HL
        LD    BC,8
        LD    HL,(59997)
        LD    DE,SALVA
        LDIR
;Se almacena el UDG original en otra direccion para protegerlo
        LD    HL,SALVA
        LD    (23728),HL
        LD    IX,(23728)
;En IX se guarda la direccion del primer dato del
;UDG protegido
        LD    HL,(59997)
;Se carga en HL el comienzo de la direccion del UDG elegido
        LD    E,0
;Se inicializa a 0 el registro E ya que va a se el
;que contendrá cada uno de los bytes resultantes
;del UDG tratado
        LD    D,8
;D=contador de los bytes del UDG1

BYTE    LD    B,8
;B=contandor de cada uno de los bits de los bytes del UDG
BUC     LD    C,(IX+0)
;Se carga en C el primer bytes a comparar y tratar
        LD    A,(59996)
        CP    0
        JR    Z,BITD
;Si se ha elegido la rotacion der/izq se salta a BITD
        JR    BITI
;Si no, a BITI
OTRA    JR    NZ,SETUP
VEZ     INC   IX
;Acceso al siguiente dato del UDG protegido
        DJNZ  BUC
;Cuando B=0 se salta a BUC para tratar otro byte
        LD    (HL),E
;Se carga en la direccion correspondiente del UDG
;el dato definitivo
        LD    IX,(23728)
;Se vuelve a reinicializar IX con la direccion
;etiquetada SALVA
        INC   HL
        DEC   D
        LD    E,0
        LD    A,D
        RET   Z
        JR    BYTE
;Salto para el siguiente dato
BITD    LD    A,D
        CP    8
        JR    Z,BIT0
        CP    7
        JR    Z,BIT1
        CP    6
        JR    Z,BIT2
        CP    5
        JR    Z,BIT3
        CP    4
        JR    Z,BIT4
```

ROTACION DE UDGs



El primer byte a comparar (1.ª fila del UDG) se carga en C utilizando IX como registro índice (línea 46) y seguidamente chequeamos el contenido de la dirección 59996 para llevar a cabo el tipo de rotación elegido. Para no alargarnos excesivamente, vamos a ver únicamente la rotación derecha-izquierda. Si se ha elegido esta última opción, el programa salta a BITD. A partir de esta etiqueta se carga en A el contenido de D, que inicialmente vale 8 (cada vez que haya sido tratada una fila del UDG, se irá decrementando). Inmediatamente se compara el contenido de este registro con valores comprendidos entre 8 y 0. En este caso, al ser el contenido de D=8, un salto condicional desvía el programa a la etiqueta BIT0 donde se com-

```
 83          CP    3
 84          JR    Z,BIT5
 85          CP    2
 86          JR    Z,BIT6
 87          CP    1
 88          JR    Z,BIT7
 89 BITI     LD    A,D
 90          CP    8
 91          JR    Z,BIT7
 92          CP    7
 93          JR    Z,BIT6
 94          CP    6
 95          JR    Z,BIT5
 96          CP    5
 97          JR    Z,BIT4
 98          CP    4
 99          JR    Z,BIT3
100          CP    3
101          JR    Z,BIT2
102          CP    2
103          JR    Z,BIT1
104          CP    1
105          JR    Z,BIT0
106 BIT0     BIT   0,C
107          JR    OTRA
108 BIT1     BIT   1,C
109          JR    OTRA
110 BIT2     BIT   2,C
111          JR    OTRA
112 BIT3     BIT   3,C
113          JR    OTRA
114 BIT4     BIT   4,C
115          JR    OTRA
116 BIT5     BIT   5,C
117          JR    OTRA
118 BIT6     BIT   6,C
119          JR    OTRA
120 BIT7     BIT   7,C
121          JR    OTRA
122 SETUP    LD    A,(59996)
123          CP    0
124          JR    Z,SETUP1
125          JR    SETUP2
126 SETUP1   LD    A,B
127          CP    8
128          JR    Z,SET7
129          CP    7
130          JR    Z,SET6
131          CP    6
132          JR    Z,SET5
```

para el contenido del bit 0 de la fila que estamos tratando. Un salto incondicional (JR OTRA) hace que nuestro programa vuelva hacia atrás a la etiqueta especificada donde nos encontramos con la instrucción JR NZ SETUP. Si el resultado anterior no ha sido cero (es decir, si el bit que hemos examinado estaba alzado) accedemos a la dirección etiquetada por SETUP (donde volvemos de nuevo a ver si estamos tratando la rotación d/i o i/d). Como hemos elegido la rotación d/i, saltamos a SETUP1 y allí chequeamos el contenido de B para saber qué número de orden del bit se quiere alterar. De esta manera, saltamos a SET7 donde se alza el primer bit del byte contenido en E (que inicialmente vale 0). De esta forma ya hemos conse-



```
133         CP   5
134         JR   Z,SET4
135         CP   4
136         JR   Z,SET3
137         CP   3
138         JR   Z,SET2
139         CP   2
140         JR   Z,SET1
141         CP   1
142         JR   Z,SET0
143 SETUP2  LD   A,B
144         CP   8
145         JR   Z,SET0
146         CP   7
147         JR   Z,SET1
148         CP   6
149         JR   Z,SET2
150         CP   5
151         JR   Z,SET3
152         CP   4
153         JR   Z,SET4
154         CP   3
155         JR   Z,SET5
156         CP   2
157         JR   Z,SET6
158         CP   1
159         JR   Z,SET7
160
161 SET7    SET  7,E
162         JP   VEZ
163 SET6    SET  6,E
164         JP   VEZ
165 SET5    SET  5,E
166         JP   VEZ
167 SET4    SET  4,E
168         JP   VEZ
169 SET3    SET  3,E
170         JP   VEZ
171 SET2    SET  2,E
172         JP   VEZ
173 SET1    SET  1,E
174         JP   VEZ
175 SET0    SET  0,E
176         JP   VEZ
177 SALVA
178 ;En las 8 direcciones sigu
ientes a SALVA se guardan
179 ;los datos del UDG que se
esta rotando
```

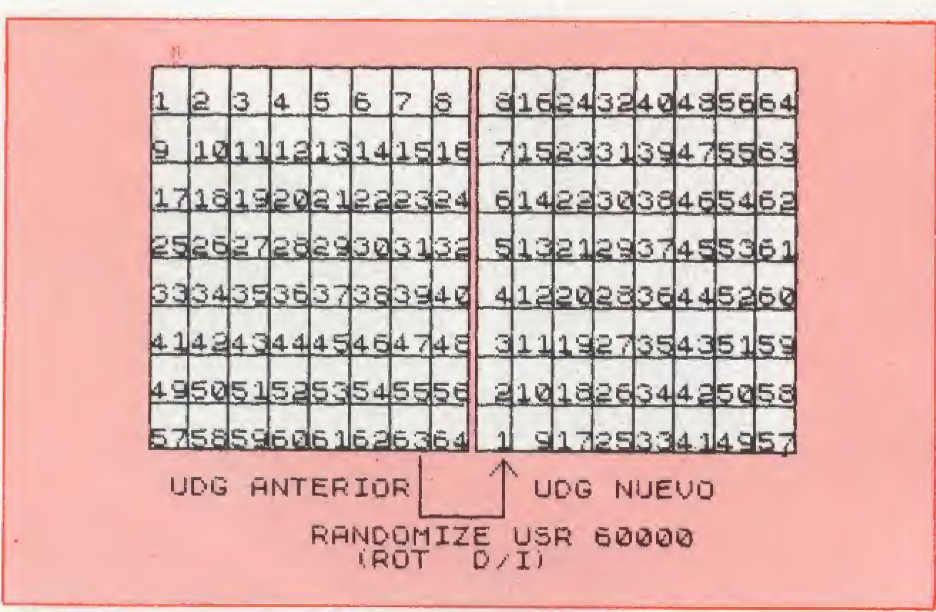


FIG. 3
(Situación de los bits tras un cuarto de giro)

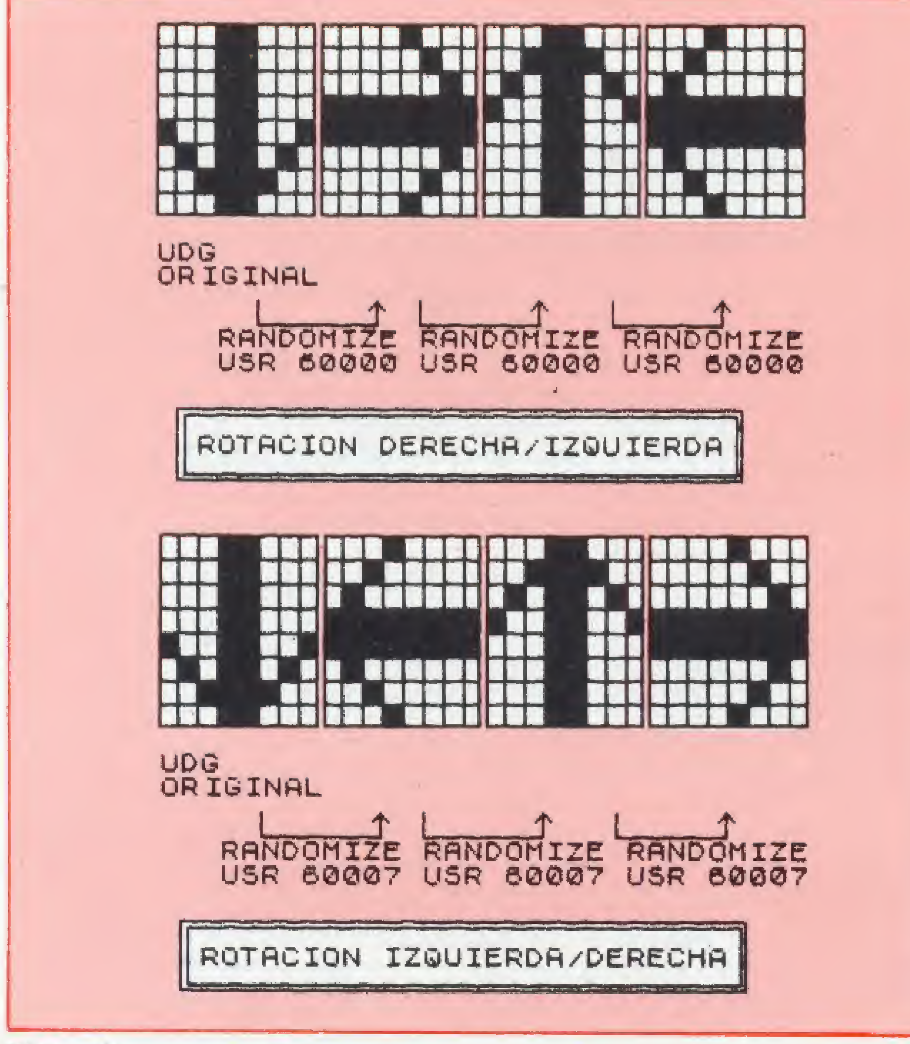


Fig. 4
(ROTACION COMPLETA DE UN UDG)

guido el primer paso; el último bit de la primera fila se traslada al lugar del 1.º bit de la primera fila.

Y el ciclo se repite

Tras esta última operación, saltamos incondicionalmente a la dirección etiquetada en VEZ donde incrementabamos el registro IX (con esto accedemos ya al siguiente Byte, de cuál tendremos que comprobar su último bit). La siguiente instrucción (DJNZ BUC) hace que se decremente B en 1 y salte a BUC mientras su valor no sea cero. A partir de BUC volvemos a repetir la misma operación anterior, cargando en C el contenido de IX + 0 que ahora ya es el 2.º dato del UDG.

Una vez que B sea igual a 0, quiere decir que hemos examiando el contenido de todos los últimos bits de las 8 filas y trasladado su contenido al registro E. Bien, pues este registro se carga en la dirección contenida en HL (línea 59) y que corresponde a la dirección inicial del UDG en esta primera parte de este complicado bucle. Volvemos a cargar en IX la dirección etiquetada en SALVA, con lo cual se accede de nuevo al primer dato del UDG (ahora se trata de comprobar el estado de cada uno de los penúltimos bits de las 8 filas del UDG). Entre las líneas 65 y 70, incrementamos HL para poder escribir sobre la siguiente dirección del UDG; decrementamos el contador de las filas (registro D), iniciamos de nuevo el registro E a 0 y si no hemos completado todas las filas, volvemos de nuevo a la etiqueta BYTE donde se reinicializa todo el proceso.

Enrevesado. ¿No? A primera vista sí lo parece, pero puede llegar a entender totalmente el proceso si sigue el programa paso a paso con un modelo de UDG como ejemplo. Entonces podrá ver que a pesar de su longitud, el programa es geniunamente sencillo, ya que no hace

más que poner en práctica los fundamentos que explicamos en nuestro apartado anterior, «Ideas básicas de la rotación».

El programa de demostración BASIC

Si no dispone de un ensamblador, debe hacer uso del programa BASIC que carga los códigos en la memoria y que al mismo tiempo ofrece una demostración de lo que es capaz de hacer.

Cuando desee utilizarla en un programa propio, deberá grabarla con SAVE "ROTACION"CODE 60000,309 y cargarla posteriormente (LOAD ""CODE). Después de haber hecho lun CLEAR 56995, antes de llamar a la rutina, se debe cargar en la dirección 59999 en n.º de Orden del gráfico a imprimir y posteriormente RANDOMIZE USR 60000 ó RANDOMIZE USR 60007 según el sentido del giro que quiera dar al gráfico. Finalmente, imprima el gráfico. El método es similar al de la ya estudiada inversión de caracteres.

Una mejora sustancial de la rutina consistiría en que rotara no solamente un gráfico, sino todo un bloque de caracteres. Aquí ya no habría que cambiar exclusivamente los bits del carácter, sino también cambiar las direcciones de los gráficos que conforman el bloque. Sobre la base de lo que ya hemos visto y si tiene los suficientes conocimientos para ello, puede intentar afrontar este nuevo problema.

Orlando Araujo Martín

```
   1 REM Orlando Araujo Martin
   2 REM Cargador ROTACION
   3 REM 13 Junio 1986
   4
   5
  50 PRINT  FLASH 1;AT 10,10;"CA
RGANDO CODIGO"
 100 LET CON=0
 120 FOR I=60000 TO 60309: READ
A
 130 POKE I,A: LET CON=CON+A: NE
XT I
 140 IF CON<>32933 THEN  PRINT "
ERROR EN DATAS": STOP
 150 REM DEMONSTRACION
 160 REM
 170 CLS
 200 LET CON=144: FOR I=144 TO 1
64: FOR N=0 TO 3: POKE 59999,144
-CON: PRINT CHR$ I;" ";: RANDOMI
ZE USR 60000: NEXT N
 210 LET CON=CON-1: PRINT : NEXT
I
 250 PRINT  INVERSE 1;AT 21,0;"D
ER./IZQ"
 300 LET X=0: LET Y=15: LET CON=
144: FOR I=144 TO 164: FOR N=0 T
O 3: POKE 59999,144-CON: PRINT A
T X,Y;CHR$ I: RANDOMIZE USR 6000
7: LET Y=Y+2: NEXT N
 310 LET CON=CON-1: LET X=X+1: L
ET Y=15: NEXT I
 350 PRINT  INVERSE 1;AT 21,15;"
IZQ./DER."
1001 DATA 62,0,50,92,234,24,5,62
,1,50,92,234,58,95,234,203,39,20
3,39,203,39,95,22,0,42
1002 DATA 123,92,25,34,93,234,1,
8,0,42,93,234,17,145,235,237,176
,33,145,235,34,176,92,221,42
1003 DATA 176,92,42,93,234,30,0,
22,8,6,8,221,78,0,58,92,234,254,
0,40,21,24,52,32,115
1004 DATA 221,35,16,238,115,221,
42,176,92,35,21,30,0,122,200,24,
223,122,254,8,40,61,254,7,40
1005 DATA 61,254,6,40,61,254,5,4
0,61,254,4,40,61,254,3,40,61,254
,2,40,61,254,1,40,61
1006 DATA 122,254,8,40,56,254,7,
40,48,254,6,40,40,254,5,40,32,25
4,4,40,24,254,3,40,16
1007 DATA 254,2,40,8,254,1,40,0,
203,65,24,167,203,73,24,163,203,
81,24,159,203,89,24,155,203
1008 DATA 97,24,151,203,105,24,1
47,203,113,24,143,203,121,24,139
,58,92,234,254,0,40,2,24,33,120
1009 DATA 254,8,40,61,254,7,40,6
2,254,6,40,63,254,5,40,64,254,4,
40,65,254,3,40,66,254
1010 DATA 2,40,67,254,1,40,68,12
0,254,8,40,63,254,7,40,54,254,6,
40,45,254,5,40,36,254
1011 DATA 4,40,27,254,3,40,18,25
4,2,40,9,254,1,40,0,203,251,195,
171,234,203,243,195,171,234
1012 DATA 203,235,195,171,234,20
3,227,195,171,234,203,219,195,17
1,234,203,211,195,171,234,203,20
3,195,171,234
1013 DATA 203,195,195,171,234,0,
0,0,0,0
```

PROGRAMA TU ORDENADOR PARA QUE JUEGUE COMO UN MAESTRO

3 EN RAYA



Aunque las reglas y estrategias del juego de las 3 en raya son muy simples, conseguir que tu ordenador juegue bien no es demasiado fácil. En este artículo enseñaremos a tu ordenador a jugar a ganar, o como mucho a empatar. Comparado con un jugador humano experimentado, la única debilidad de este programa es su ocasional pasividad; a veces se decidirá por un empate cuando es posible obtener una victoria.

Además de enseñar a tu ordenador a jugar, el programa sirve como ejemplo de tres técnicas que son igualmente aplicables para desarrollar programas de otros juegos más complejos, como el ajedrez o las damas:

- Movimientos de apertura preparados.
- «Echar un vistazo», valorar las consecuencias de un movimiento, estudiando las posibles jugadas siguientes.
- Heurística, seleccionar movimientos acordes con principios generales de estrategia.

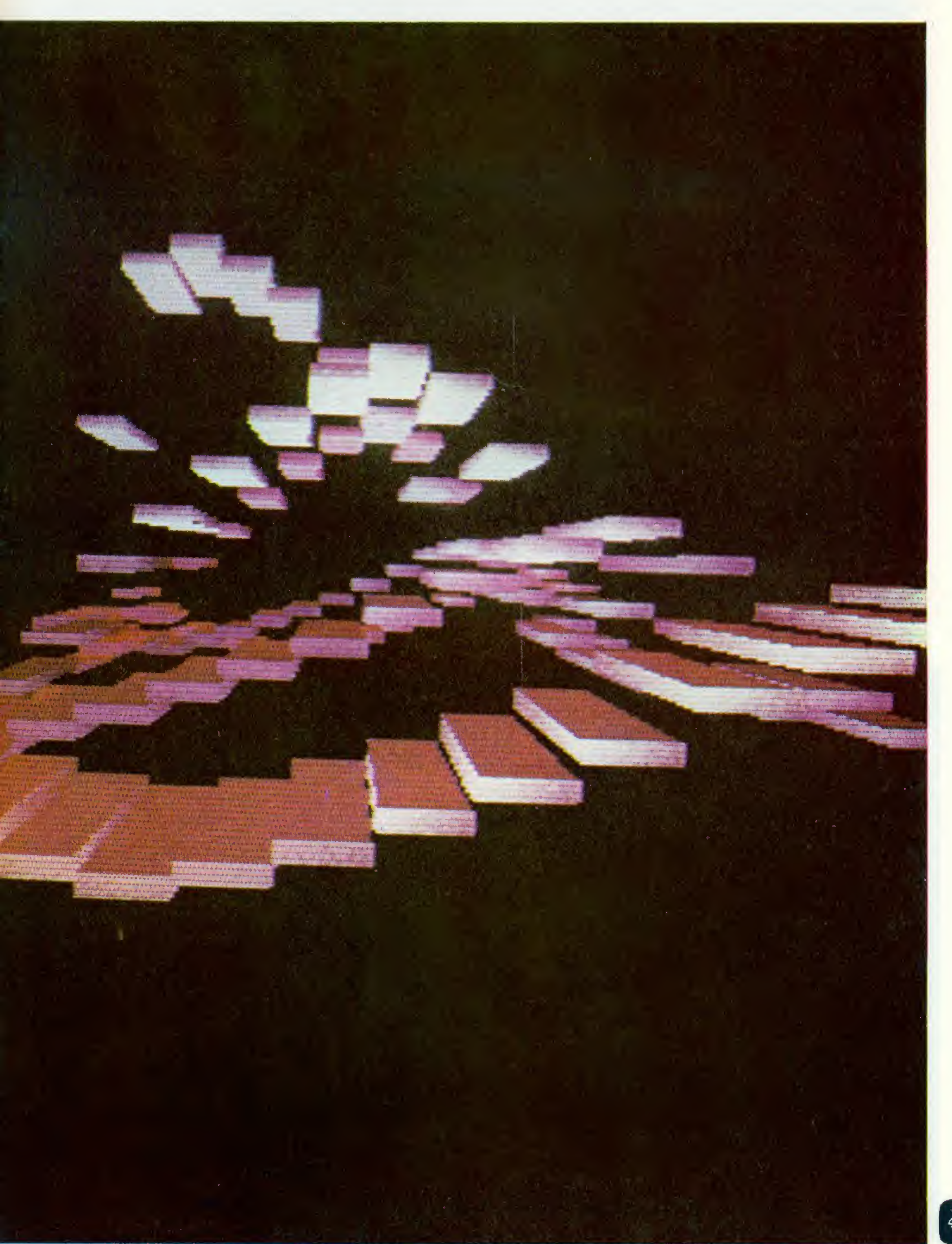
Reglas y Estrategias

Antes de explicar cómo funciona el programa, revisaremos el objeto, reglas y estrategias de este juego. Se juega en una rejilla de 3x3 casillas. Dos jugadores marcan una celda, alternatívamente. El jugador que sale, marca con una X, y el otro con O. (Nos referimos a ellos de ahora en adelante, como jugador X y jugador O).

El primer jugador que coloque tres marcas en una columna, fila o diagonal gana. Si todas las celdas están llenas sin que haya ganado ningún jugador se produce un empate. Antes de empezar los subsiguientes juegos, ambos jugadores intercambian el orden de salida, así el primero pasa a ser el segundo y viceversa. El nivel más bajo de estrategia en el juego se compone de tres pasos (1) si puedes ganar en tu siguiente turno, hazlo; (2) si tu oponente puede ganar en el suyo, bloquea su acción; (3) si no se dan ni (1), ni (2) ocupa otra casilla libre.

No lleva mucho tiempo mejorar o refinar el paso (3). La buena estrategia está basada en la trampa, una marca que te da dos oportunidades de ganar en tu siguiente jugada (gráfico 1). Tu oponente sólo será capaz de bloquear una de estas posibilidades y así cuando te toque jugar de nuevo, tendrás todavía una oportunidad de completar una raya.

Recíprocamente, para evitar per-



LISTADO 1

```
   5>RANDOMIZE
  10 CLS : POKE 23658,8: PAPER 7: INK 0: FLASH 0: INVERSE 0: BRIGHT 0
  30 DIM J$(2,8): DIM i(3,3): DIM O(9,3): DIM T(3,3): DIM J(2): DIM D(4,2): DIM
A(9,3): DIM G(2): DIM C$(3,1): DIM V(2)
  35 LET V(1)=0: LET V(2)=0: LET JE=0
  40 FOR F=1 TO 3
  50 FOR C=1 TO 3
  60 READ T(F,C)
  70 NEXT C
  80 NEXT F
  90 DATA 2,3,2,3,1,3,2,3,2
 100 FOR N=1 TO 4
 110 FOR V=1 TO 2
 120 READ D(N,V)
 130 NEXT V
 140 NEXT N
 150 DATA 0,1,1,1,1,0,1,-1
 160 FOR W=1 TO 8
 170 FOR Y=1 TO 3
 180 READ A(W,Y)
 190 NEXT Y
 200 NEXT W
 210 DATA 1,1,1,1,1,2,1,1,3,1,2,3,1,3,4,1,3,3,2,1,1,3,1,1
 230 LET C$(1)="-"
 240 LET C$(2)="X"
 250 LET C$(3)="O"
 253 LET TES=0
 254  PRINT "SI QUIERES QUE EL ORDENADOR      JUEGE SOLO,INTRODUCE '"; INK 2;"SPE
CTRON"; INK 0;"'"
 255 INPUT "CON QUIEN VOY A JUGAR "; LINE S$
 256 IF LEN S$>8 THEN  LET S$=S$(1 TO 8)
 257  LET LAR=LEN S$
 258 IF S$="SPECTRON" THEN  LET TES=1
 260 LET J$(1)=S$
 270 LET J$(2)="SPECTRUM"
 280 LET V(1)=0
 290 LET V(2)=0
 300  LET JE=0
 330  LET J(1)=1
 340 LET J(2)=2
 350 GO TO 390
```

LISTADO 2

```
 360>LET JS=J(1)
 370 LET J(1)=J(2)
 380  LET J(2)=JS
 400 CLS : PRINT AT 1,0; INK 7; PAPER 1;"          - 3 EN RAYA -          "
 405 PRINT AT 13,2;"SPECTRUM";AT 13,17-INT (LAR/2);J$(1);AT 13,25;"EMPATES"
 406 PRINT AT 8,16;"FILA  COLUMNA"
 407  PRINT AT 15,5;V(2);AT 15,17;V(1);AT 15,28;JE
 410  LET NM=0
 420 FOR F=1 TO 3
 430 FOR C=1 TO 3
 440  LET I(F,C)=0
 450 NEXT C
 460 NEXT F
```

LISTADO 3

```
470> LET NM=NM+1
480 LET NR=1
490 LET R=1
500 GO SUB 2020
520 PRINT AT 5,10;"JUEGA: "; INK J(NR)+1;J$(J(NR)); INK 0;AT 5,25;"MARCA "; INK J(NR)+1;C$(NR+1)
525  PRINT AT 10,17;"          "
530 IF J(NR)=1 THEN  GO SUB 820+180*(TES=1)
535 IF J(NR)=2 THEN  GO SUB 1000
```

LISTADO 4

```
540> LET SL=3
550 LET ST=NR
560 GO SUB 2240
580 IF N>0 THEN  GO TO 640
590 IF NM=6 THEN  GO TO 730
600 IF NM=5 THEN  GO TO 690
610  IF NR=2 THEN  GO TO 470
620  LET NR=NR+1
630 GO TO 490
640  LET P=G(N)
650 GO SUB 2020
660  PRINT AT 18,2; FLASH 1; INK 7; PAPER 1;"GANA ";J$(J(NR),1 TO 8-(J(NR)=1)*(8-LAR));"!"
670  LET V(J(NR))=V(J(NR))+1
680  GO TO 730
690  LET R=1
700  GO SUB 2020
710  PRINT AT 18,2; FLASH 1; PAPER 1; INK 7;"EMPATE"
720  LET JE=JE+1
```

LISTADO 5

```
730> PRINT AT 15,5;V(2);AT 15,17;V(1);AT 15,28;JE
740  INPUT "SEGUIMOS O LO DEJAMOS (S/D) "; LINE H$
745 PRINT AT 18,2;"                  "
750  IF H$="S" OR H$="s"  THEN  GO TO 360
760  IF H$<>"D" AND H$<>"d"  THEN  GO TO 740
810  STOP
```

LISTADO 6

```
820>PRINT AT 21,0;"FILA Y COLUMNA.(0,0) JUEGO NUEVO"
830 INPUT "FILA: ";FM;" COLUMNA: ";CM
835 PRINT AT 21,0;"                                "
840  IF FM=0 AND CM=0 THEN  GO TO 950
850  IF FM<1 OR FM>3 OR CM<1 OR CM>3 THEN  GO TO 890
860  IF I(FM,CM)=0 THEN  GO TO 980
870  PRINT #1; PAPER 1; INK 7;"        CASILLA OCUPADA        ";: PAUSE 200
880  GO TO 900
890  PRINT #1; PAPER 1; INK 7;"      NO EXISTE ESA CELDA      ";: PAUSE 200
910 LET R=1
920 GO SUB 2020
940  GO TO 820
950  PRINT #1; PAPER 1; INK 7;"        JUEGO CANCELADO        ";: PAUSE 200
960  LET NM=6: GO TO 590
970  RETURN
980 PRINT AT 10,18;FM;AT 10,25;CM: LET I(FM,CM)=NR
990  RETURN
```

der el juego, intentarás impedir que tu contrario te ponga una trampa (gráfico 1). Protegerte de las trampas no es siempre fácil. En algunos casos deberás evaluar las posibles jugadas de dos turnos sucesivos para desbaratar una posible encerrona. Incluso, la primera marca del jugador O puede conducirle a un posible fracaso. Las siete configuraciones que el jugador O debe evitar en su primer turno están en el gráfico 2. Antes de seguir leyendo, puedes comprobar por ti mismo que X puede ganar en cada una de esas situaciones.

Cómo juega el ordenador

En el siguiente párrafo, explicaremos como el ordenador maneja ambos papeles, jugador X y jugador C. Ocasionalmente, puede sonar como si el programa juega consigo mismo, pero debes tener en cuenta que en un real, tu representas un papel y la máquina otro.

Las primeras marcas de ambos jugadores son tratadas como casos especiales. El programa usa movimientos preparados para estas jugadas, sin analizar las posibles jugadas siguientes, ni utilizar métodos heurísticos. Para seleccionar las marcas de los turnos siguientes el programa utiliza cinco pruebas:

1. Buscar potenciales jugadas para ganar, marcas que completen una raya. Si encuentra varias, el programa escoge aleatoriamente entre ellas.
2. Si el programa no puede encontrar una marca que le dé la victoria, comprueba si el oponente puede ser bloqueado para prevenir una victoria en su siguiente turno (se investiga una jugada adelante). El programa bloquea la primera de esas rayas que encuentra.
3. Si el programa aún no ha marcado ninguna casilla, comienza a buscar celdas para poner una trampa al contrario, escogiendo la primera que encuentre.
4. Si ninguna de estas comprobaciones ha resultado para marcar una casilla, el programa busca casillas donde prevenir una posible trampa del contrario en su siguiente turno. Esto implica estudiar las posibles marcas de las dos jugadas siguientes.
5. Para escoger entre las celdas que pasan la prueba 4, el programa selecciona la casilla que está en las menos rayas posibles, estas rayas no deben incluir ninguna de sus propias marcas. La razón de esto, es que cuantas menos rayas haya sin marcas del jugador, menos posibilidades tiene el oponente de ganar el juego. Sin embargo, este principio no siempre produce la estrategia más agresiva, por esto el programa, ocasionalmente se colocará para forzar un empate cuando es posible obtener una victoria.

El programa

Presentamos el programa en bloques lógicos. Tecléalos según vamos avanzando en la explicación. El primer bloque inicializa el ordenador, crea varias matrices y asigna ciertos valores para algunas variables generales del programa:

ORDENADOR POPULAR

LA REVISTA QUE INTERESA TANTO AL AFICIONADO COMO AL PROFESIONAL





Una publicación que informa con amenidad acerca de las novedades en el campo de las computadoras personales.

ORDENADOR POPULAR, la revista para el aficionado a la informática.

Ya está a la venta

Cómprela en su kiosco habitual o solicítela a:

ORDENADOR POPULAR

Bravo Murillo, 377
Tel. 7339662
28020 - *MADRID*

LISTADO 7

```
1000> IF NM>1THEN GO TO 1210
1010  IF NR<>1 THEN  GO TO 1060
1020  GO SUB 2620
1030  LET FM=FT
1040  LET CM=CT
1050  GO TO 1990
1060  LET T=T(FM,CM)
1070  GO SUB 2620
1080  IF T=1 THEN  GO TO 1090
1081 IF T=2 THEN  GO TO 1110
1082 IF T=3 THEN  GO TO 1140
1090  IF I(FT,CT)=3 THEN  GO TO 1070
1100  GO TO 1180
1110  LET FT=2
1120 LET CT=2
1130 GO TO 1180
1140 IF I(FT,CT)=1 THEN  GO TO 1180
1141 IF I(FT,CT)=2 THEN  GO TO 1150
1142 IF I(FT,CT)=3 THEN  GO TO 1170
1145 LET W=FT-FM: LET Y=CT-CM
1150 IF ABS W=2 OR ABS Y=1 THEN  GO TO 1070
1160  GO TO 1180
1165 LET W=FT-FM: LET Y=CT-CM
1170  IF ABS W=1 OR ABS Y=1 THEN  GO TO 1070
1180 LET FM=FT
1190  LET CM=CT
1200 GO TO 1990
```

LISTADO 8

```
1210> IF NM>2THEN GO TO 1240
1220  IF NR=2 THEN  GO TO 1340
1230  GO TO 1500
1240  LET ST=NR
1250  LET SL=2
1260 GO SUB 2240
1270  IF N=0 THEN  GO TO 1340
1280  LET M=INT (N*RND)+1
1290  LET R=G(M)
1300 GO SUB 2480
1310  LET FM=FO
1320  LET CM=CO
1330  GO TO 1990
```

LISTADO 9

```
1340> LET ST=3-NR
1350  LET SL=2
1360 GO SUB 2240
1370  IF N=0 THEN  GO TO 1430
1380  LET R=G(N)
1390  GO SUB 2480
1400  LET FM=FO
1410 LET CM=CO
1420  GO TO 1990
```

LISTADO 1

La matriz I (fila, columna) contiene una imagen del tablero de juego. Las líneas 40 a 90 leen los tres tipos de casillas y los guardan en T(,), 1= centro, 2=esquina, 3=lado. Las líneas 100 a 150 leen en D(,) los vectores de dirección usados para generar las ocho posibles rayas que hay en el juego (gráfico 3). Para la raya R, D(R,1) es el incremento en filas y D(R,2) el incremento en columnas.

Las líneas 160 a 210 leen en A(,) los atributos de las ocho rayas. Para la raya R, A(R,1) es la fila de salida y A(R,2) es la columna, A(R,3) es la dirección que sigue.

C$() almacena los caracteres usados cuando se representa la rejilla: «—» para celda libre, «X» y «O». J$() guarda el nombre de cada jugador, J$(2) es siempre «SPECTRUM». «VC» contiene el número de victorias de cada jugador y JE el de juegos empatados, las tres son inicializadas a cero.

En la línea 254 está la rutina de introducción del nombre del jugador humano, si este nombre contiene más de 8 caracteres es truncado para que pueda almacenarse en J$(). En las líneas 330 y 340 se determinan cuáles son los jagadores para el primer enfrentamiento. J(N)=1 indica que el jugador N es humano y J(N)= 2 que es el ordenador.

El siguiente bloque de líneas se encarga de iniciar un nuevo juego:

LISTADO 2

Las líneas 360 a 380 hacen que los jugadores intercambien sus órdenes de salida cuando se inicia un nuevo juego (la línea 350 mantiene el orden para el primer juego). NM es el número de movimiento, inicialmente a 0. Un movimiento completo consiste en 2 marcas, una X y un 0. Las líneas 420 a 460 vacían la rejilla para un nuevo juego. Ahora el programa está preparado para la primera marca:

LISTADO 3

Primero el contador de movimientos es incrementado y el número de jugadas es 1 (líneas 470 a 480). La subrutina llamada en 500

LISTADO 10

```
1430> IF NM=2THEN GO TO 1500
1440  LET ST=NR
1450 GO SUB 2660
1460  IF N<>2 THEN  GO TO 1500
1470  LET FM=FV
1480 · LET CM=CV
1490  GO TO 1990
```

LISTADO 11

```
1500> LET FS=0
1510  FOR F=1 TO 3
1520 FOR C=1 TO 3
1530  IF I(F,C)<>0 THEN  GO TO 1760
1540  LET I(F,C)=NR
1550  LET ST=NR
1560  LET SL=2
1570  GO SUB 2240
1580  IF N=0 THEN  GO TO 1680
1590  IF NM=2 AND NR=1 THEN  GO TO 1720
1600  LET R=G(1)
1610  GO SUB 2480
1620  LET ST=3-NR
1630  LET I(FO,CO)=ST
1640  LET SL=2
1650  GO SUB 2240
1660  LET I(FO,CO)=0
1670  GO TO 1710
1680  IF NM=2 AND NR=1 THEN  GO TO 1750
1690  LET ST=3-NR
1700 GO SUB 2660
1710  IF N=2 THEN  GO TO 1750
1720  LET FS=FS+1
1730  LET O(FS,1)=F
1740  LET O(FS,2)=C
1750  LET I(F,C)=0
1760  NEXT C
1770  NEXT F
```

LISTADO 12

```
1780> LET SL=2
1790  LET ST=3-NR
1800  FOR C=1 TO F
1810  LET I(O(C,1),O(C,2))=NR
1820  GO SUB 2240
1830  LET I(O(C,1),O(C,2))=0
1840  LET O(C,3)=M
1850  NEXT C
1860 IF FS<>1 THEN  GO TO 1890
1870  LET CN=1
1890  LET SM=1
1900  FOR W=2 TO FS
1910  IF O(SM,3)<O(W,3) THEN  GO TO 1930
1920  LET SM=W
1930  NEXT W
1940  LET CN=INT (FS*RND)+1
1950  IF O(CN,3)=O(SM,3) THEN  GO TO 1970
1960  GO TO 1940
1970  LET FM=O(CN,1)
1980  LET CM=O(CN,2)
```

imprime la rejilla de juego en el estado actual y la 520 indica de quién es el turno de jugar. Dependiendo del jugador que debe marcar, las líneas 530 y 535 llaman a una u otra subrutina. La de la línea 820 es para recibir desde teclado la elección de casilla del jugador humano, y la de la línea 1000 para que el programa siguiendo su lógica, seleccione una celda. Después de realizar la selección el programa evalúa su efecto:

LISTADO 4

La subrutina llamada en 560 busca en las ocho rayas para ver si el jugador actual ha ganado. N>0 indica que ha ganado, en ese caso las líneas 640 a 680 anuncian el nombre del jugador e incrementan su contador de victorias. Si N=0, el ordenador comprueba el número de movimientos NM para ver si el juego ha terminado de alguna otra manera. Hay sólo 9 casillas en la rejilla, y cada movimiento representa ocupar dos de éstas, una para cada jugador. Ordinariamente, NM no puede exceder nunca de 5, porque la X del movimiento número 5 siempre ocupa la novena casilla (2+2+2+2+1=9 casillas). Sin embargo, si un jugador cancela el juego, NM de carga con 6. La línea 590 detecta esa condición y salta al menú de continuación.

Cuando NM=5, el ordenador deduce un empate (no hay ganador y todas las celdas están marcadas), y lo anuncia (líneas 690 a 720). En otro caso NM es menor que 5, así el programa da el turno al siguiente jugador.

Las líneas siguientes imprimen el menú de continuación al final del juego:

LSITADO 5

Se imprimen los contadores de victorias y de empates y se pregunta si se quiere terminar o seguir jugando. Este es el final del programa principal. Ahora presentaremos las dos rutinas más grandes, el turno del humano y el turno del ordenador.

Subrutinas

El programa llama a la siguiente

LISTADO 13

```
1990> LET I(FM,CM)=NR
2000  PRINT AT 10,18;FM;AT 10,25;CM
2010 RETURN
```

LISTADO 14

```
2020> LET QF=A(R,1)
2030  LET QC=A(R,2)
2040  LET DN=A(R,3)
2050  LET QL=0
2070  FOR W=1 TO 3
2080  PRINT AT 5+(W-1)*2,1;"";
2090 FOR Y=1 TO 3
2100  IF QL=3 OR W<>QF OR Y<>QC THEN
      GO TO 2180
2120  PRINT C$(I(W,Y)+1);
2140  LET QF=QF+D(DN,1)
2150 LET QC=QC+D(DN,2)
2160  LET QL=QL+1
2170  GO TO 2190
2180  PRINT C$(I(W,Y)+1);
2190  PRINT " ";
2200  NEXT Y
2220  NEXT W
2230  RETURN
```

LISTADO 15

```
2240> LET N=0
2250 LET M=0
2260  FOR R=1 TO 8
2270  LET FU=A(R,1)
2280  LET CU=A(R,2)
2290  LET DN=A(R,3)
2300  LET NF=0
2310 LET MF=0
2320  FOR Z=1 TO 3
2330  IF I(FU,CU)=0 THEN  GO TO 2380
2340  IF I(FU,CU)=ST THEN  GO TO 2370
2350  LET MF=MF+1
2360 GO TO 2380
2370  LET NF=NF+1
2380  LET FU=FU+D(DN,1)
2390  LET CU=CU+D(DN,2)
2400 NEXT Z
2410  IF NF<>SL OR MF>0 THEN  GO TO 2440
2420  LET N=N+1
2430  LET G(N)=R
2440  IF MF>0 THEN  GO TO 2460
2450  LET M=M+1
2460  NEXT R
2470 RETURN
```

subrutina cuando es el turno del jugador humano:

LISTADO 6

Las líneas 820 a 830 inquieren al jugador para que especifique la fila y columna de la celda a marcar. Las filas están numeradas de arriba a abajo y las columnas de izquierda a derecha. Si el jugador selecciona 0,0, el juego es cancelado. Cualquier otro par fila-columna inválido detiene el programa y pregunta de nuevo al jugador su elección (870-940).

Dado un par fila-columna válido, la línea 860 determina si la celda está vacía, I(FM,CM)=0. Si la celda está vacía la línea 980 imprime la opción y la marca en la matriz imagen. La línea 990 retorna al programa principal.

Cuando es el turno del ordenador, el programa llama a la rutina en la línea 1000. El programa usa movimientos preparados para la primera X y el primer O. La primera X es una elección aleatoria, y el primero 0 está determinado por la posición de la X anterior. Para los movimientos siguientes, el ordenador usa la lógica de evaluar los posibles movimientos sucesivos.

Las siguientes líneas manejan las primeras X y O del ordenador:

LISTADO 7

Si el número de movimiento NM es mayor que 1, la línea 1000 hace que el programa salte a las rutinas de investigación de posibles jugadas sucesivas que serán descritas más adelante. Si NM=1 y NR=1, el programa hace la primera marca, una X. La subrutina llamada en la línea 1020 selecciona una celda aleatoriamente, y las líneas 1030 y 1040 guardan la situación de dicha casilla en las variables FM y CM. La línea 1050 provoca un salto al final de la subrutina que maneja el turno del ordenador.

Las líneas 1060 a 1200 actúan cuando el ordenador es el jugador O y el número de movimiento es 1; deben encontrar una celda que evite las siete posiciones de pérdida del juego. En la línea 1060, las variables FM y CM contienen la fila y columna del movimiento más reciente; en

otras palabras, informan al programa cuál celda contiene la X. La línea 1060 determina en qué tipo de casilla está la X: centro, esquina o lateral.

La subrutina llamada en la línea 1070 selecciona al azar una celda vacía T(FT,CT) como candidata para el siguiente movimiento del jugador O. Las 1080, 1081, 1082 saltan a la comprobación de seguridad apropiada, dependiendo del tipo de celda que está marcada con X.

Una vez que el programa ha localizado una celda segura, las líneas 1180 y 1190 guardan su dirección de fila y columna, y la línea 1200 salta al final de la subrutina del turno del ordenador.

En el caso del segundo movimiento y los siguientes, el programa no usará movimientos preparados. Primero comprobará si puede ganar con una marca:

LISTADO 8

Para los números de movimiento, NM, iguales o mayores que 3, las líneas 1240 a 1330 buscan una casilla que de la victoria. La subrutina llamada en la línea 1260 cuenta el número de rayas no bloqueadas que contengan al menos 2 marcas dle jugador NR. Si N=0, no hay tales rayas, así el programa salta a la rutina del movimiento defensivo.

Si N>0, entonces la matriz G() lista las rayas que contienen celdas ganadoras. La línea 1280 selecciona aleatoriamente una de estas rayas, y la subrutina llamada en la línea 1300 encuentra la fila y columna de la celda libre en esa raya. En la lınea 1280 se genera un número aleatorio entre 1 y N.

Cuando el programa localiza una celda para ganar, las líneas 1310 y 1320 almacenan su número de fila y de columna y la línea 1330 salta al final de la subrutina. Si el programa no puede encontrar una celda ganadora, comprueba si puede impedir que su oponente gane en su siguiente turno:

LISTADO 9

La línea 1340 sitúa en ST el número del jugador contrario. La subrutina llamada en 1360 cuenta el número de rayas no bloqueadas que contienen al menos 2 marcas del jugador contrario. Si N=0, no hay rayas de ese tipo, y el programa salta a la rutina de colocación de trampas. Si N no es 0, entonces hay al menos una posibilidad de que el contrario gane en su siguiente marca. Las líneas 1380 y 1390 encuentran la celda ganadora del oponente y las líneas 1400 y 1410 almacenan la dirección de su fila y su columna, y así el ordenador puede ocuparla. La línea 1420 salta al final de la rutina del turno de la máquina.

LISTADO 10

La subrutina de la línea 2660 comprueba cada casilla vacía para ver cuál, si hay alguna, produce una trampa. Si N=2, el programa ha encontrado una celda, y las líneas 1470 y 1480 guardarán su número de fila y columna para que el ordenador la utilice. La línea 1490 salta al final de la subrutina del turno del ordenador.

Si no se encuentran oportunidades de colocar una trampa, el programa comprobará cada celda vacía para ver cuál impediría al contrario colocar una trampa en su siguiente marca. Esta es la evaluación de posibles jugadas futuras de más alcance que realiza el programa:

LISTADO 11

La variable FS cuenta el número de celdas seguras (aquellas que impiden al oponente poner una trampa). En las líneas 1730 y 1740, para cada celda segura FS que es encontrada, O(FS,1) y O(FS,2) guardan su número de fila y columna. Una vez que el programa ha localizado todas las celdas seguras, aplica el método heurístico para escoger entre ellas:

LISTADO 12

El programa marca cada celda segura (línea 1810) y entonces cuenta cuántas rayas M no bloqueadas quedan. Para cada celda segura FS, P(FS,3) guarda el número de rayas no bloqueadas que quedan cuando la celda es marcada.

Las líneas 1900 a 1930 comparan los resultados de estas marcas de prueba para ver cuáles de éstas están en el menor número posible (SM) de rayas no bloqueadas. En las líneas 1940 a 1960 el programa escoge aleatoriamente celdas seguras hasta que encuentra una que deje SM rayas no bloqueadas. En la lınea 1940 se genera un número entre 1 y FS.

Las siguientes líneas finalizan la subrutina que controla el turno del ordenador:

LISTADO 13

La línea 1990 marca el número de jugador NR en la posición de rejilla I(FM,CM). La línea 2000 anuncia el movimiento, y la línea 2010 devuelve el control al programa principal.

Rutina de impresión y auxiliares

Aquí está la rutina que imprime el tablero de juego:

LISTADO 14

La línea 2080 controla la línea de impresión, la columna se controla imprimiendo espacios (línea 2190).

Las rutinas auxiliares son requeridas en ciertas secciones del programa para realizar funciones menores. Esta primera subrutina analiza el contenido de las seis rayas posibles:

LISTADO 15

Al volver de esta subrutina, la matriz G() lista los números de las N rayas no bloqueadas que contengan al menos SL marcas del jugador ST. La variable N cuenta el número de rayas no bloqueadas que contienen al menos SL marcas del jugador ST. La variable M cuenta el número de rayas que no contienen ninguna marca del otro jugador.

La rutina que viene a continuación localiza la primera marca en la raya R:

LISTADO 16

De vuelta, FO es la fila de la raya abierta y CO la columna. La siguiente se encarga de seleccionar una casilla vacía:

LISTADO 18

```
2660> FOR W=1TO 3
2670 FOR Y=1 TO 3
2680 IF I(W,Y)<>0 THEN  GO TO 2780
2690 LET SL=2
2700  LET I(W,Y)=ST
2710  GO SUB 2240
2720  LET I(W,Y)=0
2730  IF N<2 THEN  GO TO 2780
2740  LET FV=W
2750 LET CV=Y
2760  LET W=3
2770  LET Y=3
2780  NEXT Y
2790 NEXT W
2800 RETURN
```

LISTADO 17

```
2620> LET FT=INT (3*RND)+1
2630 LET CT=INT (3*RND)+1
2640  IF I(FT,CT)<>0 THEN  GO TO 2620
2650 RETURN
```

LISTADO 16

```
2480> LET FO=0
2490 LET CO=0
2500  LET FT=A(R,1)
2510 LET CT=A(R,2)
2520  LET DN=A(R,3)
2530  FOR W=1 TO 3
2540  IF I(FT,CT)<>0 THEN  GO TO 2580
2550  LET FO=FT
2560  LET CO=CT
2570  LET W=3
2580  LET FT=FT+D(DN,1)
2590  LET CT=CT+D(DN,2)
2600  NEXT W
2610 RETURN
```

LISTADO 17

En las líneas 2620 y 2630 se obtienen el número de fila y columna aleatoriamente. En el retorno, la fila se guarda en FT y la columna en CT. Por último, presentamos la subrutina que busca la oportunidad de colocar una trampa:

LISTADO 18

Al principio de la subrutina, ST es el número de jugador que intenta colocar la trampa. Al final, N=2 indica que se ha encontrado una trampa y, FV y CV almacenan el número de fila y columna de la celda donde se va a colocar.

Retoques

Puedes conseguir que la máquina juege consigo misma si en la petición de nombre introduces "SPECTRON". En ese caso, un testigo (TES) se ponee a 1 en la línea 530 que provoca el salto a la rutina de petición de celda para el jugador humano, se fuerza con una condición lógica el salto a la rutina de turno del ordenador, sólo que ahora el número de jugador es 1 y la lógica del programa opera con las otras fichas siguiendo la misma estrategia descrita al principio del artículo.

Después de jugar contra la máquina puedes encontrar frustrante que el ordenador nunca pierda. Si juegas tan bien como él, cada juego acabará en empate. Para hacerlo más excitante, puedes simplificar la estrategia del ordenador de varias maneras. Primero, puedes eliminar los movimientos preparados para el primer O, saltando esa sección del logical. Por ejemplo:

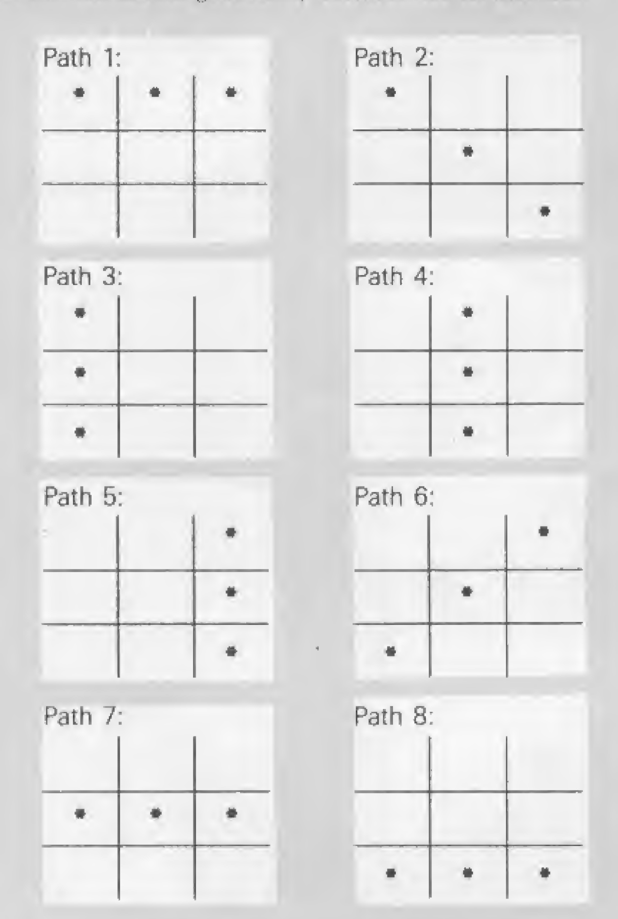
```
1010 REM ANTES HABIA: IF
     NR<>1 THEN GOTO 1060
```

Después de hacer este cambio, observarás que a menudo, el orde-

O	O	•
	X	
X	X	•

X		•
	X	
•		O

El jugador O ha caído en la trampa. El jugador X puede ganar en su próximo turno pues tiene dos marcas ganadoras, indicadas con un asterisco.



El jugador O puede evitar la trampa marcando cualquiera de las dos casillas seguras indicadas con asterisco.

X CENTER,
O ANY SIDE:

	X	
	O	

X SIDE,
O FAR CORNER:

		O
X		

Las siete posiciones de fracaso para el jugador O en su primera marca.

X SIDE,
O NEAR SIDE:

	O	
X		

X CORNER,
O NEAR SIDE:

X		
O		

X CORNER,
O NEAR CORNER:

X		
O		

X CORNER,
O FAR SIDE:

X		
	O	

X CORNER,
O FAR CORNER:

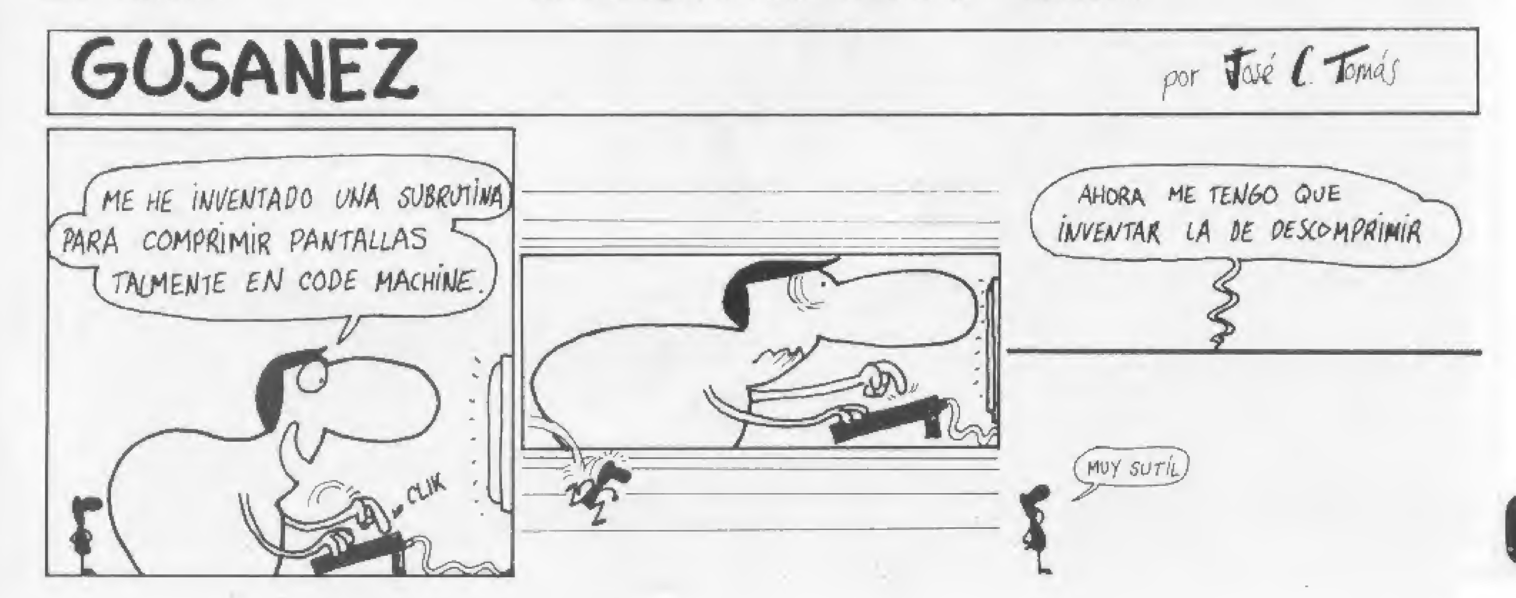
X		
		O

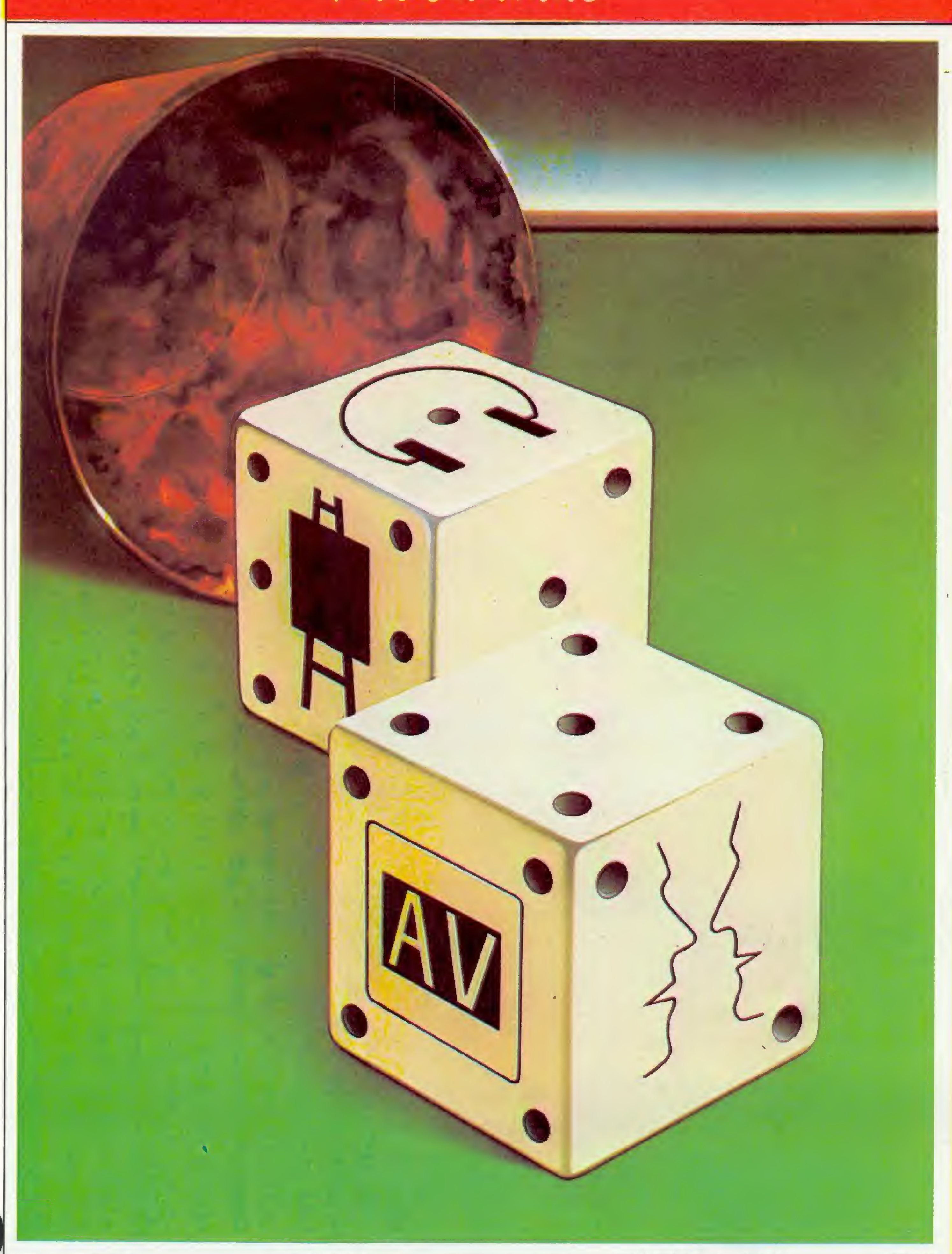
Las ocho posibles rayas en una rejilla de «3 en raya».

nador mueve a las posiciones de perdición del gráfico 2. Puedes también eliminar la posibilidad de que el ordenador ponga o detecte trampas mediante:

```
1500 GO TO 1020: REM ANTES
     HABIA: LET FS=0
```

Después de hacer este cambio, el ordenador jugará al nivel más bajo de estrategia, comentado en el párrafo "COMO JUEGA EL ORDENADOR". Incluso en este nivel la máquina puede sorprenderte realizando un brillante movimiento aleatorio.




AV

CRAPS

Basado en el juego de dados «Craps», el programa que nos ha enviado desde Valencia José Gallego muestra en todo momento información sobre la cantidad apostada, el dinero de que se dispone y el récord de ganancias. Incluye unas completas instrucciones, aunque su manejo no es nada complicado, ya que sólo hay que responder a las preguntas que efectúa el ordenador. Las líneas 5 y 6 permiten utilizar un juego de caracteres diferente al del Spectrum, situado a partir de la dirección 64250. Por consiguiente, si no se va a emplear un juego de caracteres alternativo, es necesario eliminarlas.

```
   1 REM
                C R A P S

              JUEGO DE DADOS

   2 REM
   3 REM JOSE GALLEGO GONZALEZ
           VALENCIA-ABRIL 1986
   4 REM
   5 LOAD "chars"CODE 64250,768
   6 POKE 23606,250: POKE 23607,
249
   8 CLS : PRINT  FLASH 1;AT 11,
10;"PARA LA CINTA"
  10 GO SUB 9000
  15 POKE 23658,8: LET di=100: L
ET rc=0: LET n$="******"
  20 REM PRESENTACION
  21 BORDER 4: PAPER 4: INK 0: C
LS
  24 FOR x=7 TO 0 STEP -1
  30 INK X
  32 PRINT AT 8,4;"
                  "
  34 PAUSE 5
  36 PRINT AT 9,4;"
                  "
  38 PAUSE 5
  40 PRINT AT 10,4;"
                  "
  42 PAUSE 5
  44 PRINT AT 11,4;"
                  "
  46 PAUSE 5
  48 PRINT AT 12,4;"
                  "
  50 PAUSE 5
  52 NEXT x
  55 PRINT  PAPER 7; INK 0;AT 3,
5;"AB";AT 4,5;"CD"
  58 FOR x=1 TO 100: NEXT x
  60 PRINT  PAPER 7; INK 0;AT 1,
15;"EE";AT 2,15;"EE"
  62 FOR x=1 TO 100: NEXT x
  65 PRINT  PAPER 7; INK 0;AT 3,
25;"FB";AT 4,25;"CG"
  67 FOR x=1 TO 100: NEXT x
  70 PRINT  PAPER 7; INK 0;AT 16
,28;"FH";AT 17,28;"IG"
  72 FOR x=1 TO 100: NEXT x
  75 PRINT  PAPER 7; INK 0;AT 16
,15;"E ";AT 17,15;" E"
  77 FOR x=1 TO 100: NEXT x
  80 PRINT  PAPER 7; INK 0;AT 16
,2;"JJ";AT 17,2;"KK"
  88 FOR x=1 TO 100: NEXT x
  90 FOR I=0 TO 250
  92 POKE 23606,I
  94 PRINT #1;AT 1,7; JOSE GAL
LEGO 1986"
  96 NEXT I
  98 FOR x=1 TO 500: NEXT x
```

```
 100 REM INSTRUCCIONES
 105 CLS
 110 PRINT AT 1,7;"* INSTRUCCION
ES *"
 115 PRINT AT 3,0;" El juego con
siste en lanzar losdados y jugar
 contra la banca."
 120 PRINT AT 6,0;" Si sale 7 o
11 a la primera ti-rada, se gana
 la misma  cantidadde lo apostad
o."
 125 PRINT AT 10,0;" Si sale  un
 2, 3 o 12 a la pri- mera, se pi
erde."
 130 PRINT AT 13,0;" Cualquier
otra  puntuacion, seconvierte en
 TU PUNTO. Si siguestirando y re
pites  tu punto, ga-nas el doble
 de lo apostado,perosi sale 7, p
ierdes."
 135 PRINT #0;AT 1,0;"PULSA PARA
 EMPEZAR"
 140 PAUSE 0
 145 RANDOMIZE
 150 LET di=100: CLS
 155 INPUT #1;AT 1,0;"`Tu nombre
? (max.6 letras)"; LINE a$
 160 IF LEN a$<1 OR LEN a$>6 THE
N  GO TO 155
 200 REM JUEGO
 202 LET ap=0: LET ti=0: LET tp=
0
 205 CLS
 210 PRINT  INK 0;"DINERO";TAB (
9);"APUESTA";TAB (21); FLASH 0;"
RECORD"
 215 PRINT  INK 0;TAB (1);di;TAB
 (11);ap;TAB (19);n$;TAB (27);rc
 225 INPUT #1;AT 1,0;"`Cuanto ap
uestas? (min.10)";ap
 227 IF ap<10 OR INT ap<>ap THEN
  GO TO 225
 230 IF ap<=di THEN  GO TO 240
 235 PRINT #1;AT 1,0;"    @NO TI
ENES TANTO DINERO!": BEEP 1,-10:
 FOR x=1 TO 200: NEXT x: GO TO 2
25
 240 LET di=di-ap: INPUT 0
 245 PRINT  INK 0;AT 1,1;"
";AT 1,1;di;AT 1,10;"       ";AT
1,11;ap
 247 PRINT #1;AT 1,8;"PULSA PARA
 TIRAR": IF INKEY$="" THEN  GO T
O 247
 248 PAUSE 0: INPUT 0: PAUSE 10
 250 PRINT AT 7,12;"  ";AT 8,12;
"  ";AT 5,16;"  ";AT 6,16;"  "
 252 FOR n=5+(RND*15) TO 0 STEP
-1
 253 BEEP .005,10
 254 PAUSE 6: NEXT n
 256 PRINT AT 7,12;"  ";AT 8,12;
"  ";AT 5,16;"  ";AT 6,16;"  "
 258 LET c1=0: LET c2=0: LET x1=
19: LET x2=17: LET t1=0: LET t2=
0: LET ti=ti+1: LET re=0
 260 IF c1=7 AND c2=7 THEN  GO T
O 280
 270 PRINT AT x1,12;"  ";AT x1+1
,12;"  ";AT x2,16;"  ";AT x2+1,1
6;"  "
 280 LET a1=0: LET a2=0
 290 IF c1>=1 THEN  LET x1=x1-2
 300 IF c2>=1 THEN  LET x2=x2-2
 305 IF c1>=7 AND c2>=7 THEN  GO
 TO 600
 310 LET a=INT (RND*6+1)
 320 LET b=INT (RND*6+1)
 330 LET c1=c1+1: LET c2=c2+1
 340 LET a1=a*100+900
 350 IF c1<=7 THEN  GO SUB a1
 360 LET t1=t1+2
 370 LET a2=b*100+1900
 580 IF c2<=7 THEN  GO SUB a2
 590 LET t2=t2+2
 595 GO TO 260
 600 REM RESULTADO TIRADA
 602 LET re=a+b
 605 IF re=2 THEN  LET r$="DOS"
 606 IF re=3 THEN  LET r$="TRES"
 607 IF re=4 THEN  LET r$="CUATR
O"
 608 IF re=5 THEN  LET r$="CINCO
"
 609 IF re=6 THEN  LET r$="SEIS"
 610 IF re=8 THEN  LET r$="OCHO"
 611 IF re=9 THEN  LET r$="NUEVE
"
 612 IF re=10 THEN  LET r$="DIEZ
"
```

```
 613 IF re=11 THEN  LET r$="ONCE
"
 614 IF re=12 THEN  LET r$="DOCE
"
 615 IF re=tp AND ti>1 THEN  GO
TO 800
 620 IF (re=7 OR re=11) AND ti<=
1 THEN  GO TO 700
 630 IF (re=2 OR re=3 OR re=12)
AND ti<=1 THEN  GO TO 730
 640 IF (re=4 OR re=5 OR re=6 OR
 re=8 OR re=9 OR re=10) AND ti<=
1 THEN  GO TO 760
 650 IF re=7 AND ti>1 THEN  GO T
O 730
 660 IF re<>tp AND ti>1 THEN  GO
 TO 830
 700 REM PREMIO
 701 IF re=7 THEN  LET r$="@SIET
E!"
 702 IF re=11 THEN  LET r$="@ONC
E!"
 705 BEEP 0.5,20
 710 PRINT  INK 0;AT 14,13;r$;AT
 16,10;"HAS GANADO ";ap: LET di=
di+ap*2
 715 IF di>100 AND di>rc THEN  L
ET rc=di: LET n$=a$: PRINT AT 1,
19;"       ": PRINT  INK 0;AT 1,1
9;n$;AT 1,27;rc: PRINT  INK 0; F
LASH 1;AT 0,21;"RECORD"
 720 FOR x=1 TO 300: NEXT x: GO
SUB 3000
 725 GO TO 200
 730 IF re=2 THEN  LET r$="DOS"
 731 IF re=3 THEN  LET r$="TRES"
 732 IF re=12 THEN  LET r$="DOCE
"
 733 IF re=7 THEN  LET r$="SIETE
"
 735 BEEP 0.5,-10
 740 PRINT  INK 0;AT 14,13;r$;AT
```

* INSTRUCCIONES *

El juego consiste en lanzar los dados y jugar contra la banca.

Si sale 7 o 11 a la primera tirada, se gana la misma cantidad de lo apostado.

Si sale un 2, 3 o 12 a la primera, se pierde.

Cualquier otra puntuacion, se convierte en TU PUNTO. Si sigues tirando y repites tu punto, ganas el doble de lo apostado,pero si sale 7, pierdes.

DINERO 10 APUESTA 720 RECORD 2170

TU PUNTO 8

¡TU PUNTO!

HAS GANADO 1440

```
16,9;"HAS PERDIDO ";ap
 746 IF di<10 THEN  GO TO 3500
 750 FOR x=1 TO 300: NEXT x: GO
SUB 3000
 755 GO TO 200
 770 PRINT  INK 0;AT 14,13;r$;AT
 16,12;"TU PUNTO"
 772 FOR x=1 TO 200: NEXT x
 775 PRINT  INK 0;AT 9,0;"TU PUN
TO";AT 11,3;re: LET tp=re
 780 FOR x=1 TO 100: NEXT x
 782 PRINT AT 14,13;"         ";AT
 16,12;"           "
 790 GO TO 247
 800  BEEP 0.5,20
 810 PRINT  INK 0;AT 14,12;"@TU
PUNTO!";AT 16,10;"HAS GANADO ";a
p*2: LET di=di+ap*3
 815 IF di>100 AND di>rc THEN  L
ET rc=di: LET n$=a$: PRINT AT 1,
19;"          ": PRINT  INK 0;AT 1,1
9;n$;AT 1,27;rc: PRINT  INK 0; F
LASH 1;AT 0,21;"RECORD"
 820 FOR x=1 TO 300: NEXT x: GO
SUB 3000
 825 GO TO 200
 840 PRINT  INK 0;AT 14,13;r$
 844 FOR x=1 TO 200: NEXT x
 846 PRINT AT 14,13;"         "
 850 GO TO 247
1000 REM CARAS DADO 1
1001 PRINT  PAPER 7; INK 0;AT x1
,12;"AB";AT x1+1,12;"CD"
1010 BEEP 0.01,1
1020 FOR x=1 TO t1: NEXT x
1040 RETURN
1100 PRINT  PAPER 7; INK 0;AT x1
,12;"E ";AT x1+1,12;" E"
1110 BEEP 0.01,1
1120 FOR x=1 TO t1: NEXT x
1140 RETURN
1200 PRINT  PAPER 7; INK 0;AT x1
,12;"FB";AT x1+1,12;"CG"
1210 BEEP 0.01,1
1220 FOR x=1 TO t1: NEXT x
1240 RETURN
1300 PRINT  PAPER 7; INK 0;AT x1
,12;"EE";AT x1+1,12;"EE"
1310 BEEP 0.01,1
1320 FOR x=1 TO t1: NEXT x
1340 RETURN
1400 PRINT  PAPER 7; INK 0;AT x1
,12;"FH";AT x1+1,12;"IG"
1410 BEEP 0.01,1
1420 FOR x=1 TO t1: NEXT x
1440 RETURN
1500 PRINT  PAPER 7; INK 0;AT x1
,12;"JJ";AT x1+1,12;"KK"
1510 BEEP 0.01,1
1520 FOR x=1 TO t1: NEXT x
1540 RETURN
2000 REM CARAS DADO 2
2001 PRINT  PAPER 7; INK 0;AT x2
,16;"AB";AT x2+1,16;"CD"
2010 BEEP 0.01,1
2020 FOR x=1 TO t2: NEXT x
```

```
2040 RETURN
2100 PRINT  PAPER 7; INK 0;AT x2
,16;"E ";AT x2+1,16;" E"
2110 BEEP 0.01,1
2120 FOR x=1 TO t2: NEXT x
2140 RETURN
2200 PRINT  PAPER 7: INK 0;AT x2
,16;"FB";AT x2+1,16;"CG"
2210 BEEP 0.01,1
2220 FOR x=1 TO t2: NEXT x
2240 RETURN
2300 PRINT  PAPER 7; INK 0;AT x2
,16;"EE";AT x2+1,16;"EE"
2310 BEEP 0.01,1
2320 FOR x=1 TO t2: NEXT x
2340 RETURN
2400 PRINT  PAPER 7; INK 0;AT x2
,16;"FH";AT x2+1,16;"IG"
2410 BEEP 0.01,1
2420 FOR x=1 TO t2: NEXT x
2440 RETURN
2500 PRINT  PAPER 7; INK 0;AT x2
,16;"JJ";AT x2+1,16;"KK"
2510 BEEP 0.01,1
2520 FOR x=1 TO t2: NEXT x
2540 RETURN
3000 REM FIN DE JUEGO
3001 POKE 23658,8
3003 PRINT AT 9,0;"           ";AT
11,3;"  ";AT 14,12;"
";AT 16,9;"                       "
3005 PRINT #1;AT 1,0;"`SIGUES JU
GANDO?(S/N)"
3010 IF INKEY$="" OR (INKEY$<>"S
" AND INKEY$<>"N") THEN  GO TO 3
010
3020 IF INKEY$="S" THEN  RETURN
3030 IF INKEY$="N" THEN  PRINT #
1;AT 1,0;"`JUEGA OTRO?(S/N)     "
3035 PAUSE 0
3040 IF INKEY$="" OR (INKEY$<>"S
" AND INKEY$<>"N") THEN  GO TO 3
040
3045 IF INKEY$="N" THEN  GO TO 3
100
3050 IF INKEY$="S" THEN  PRINT #
1;AT 1,0;"`QUIERES INSTRUCCIONES
?(S/N)"
3055 PAUSE 0
3060 IF INKEY$="" OR (INKEY$<>"S
" AND INKEY$<>"N") THEN  GO TO 3
060
3065 IF INKEY$="S" THEN  GO TO 1
00
3070 IF INKEY$="N" THEN  GO TO 1
45
3100 POKE 23606,0: POKE 23607,60
: BORDER 7: PAPER 0: CLS : PAUSE
 30: PAPER 7: CLS : PRINT #0;
1982 Sinclair Research Ltd.": PA
USE 0: GO TO 9999
3500 FOR x=1 TO 300: NEXT x
3503 PRINT AT 9,0;"           ";AT
11,3;"  ";AT 14,12;"
";AT 16,9;"                       "
3505 PRINT #1;AT 1,5;"NO TIENES
DINERO PARA            SEGUIR A
POSTANDO"
3510 FOR x=1 TO 300: NEXT x
3515 INPUT 0
3520 PRINT #1;AT 1,0;"`JUEGA OTR
O?(S/N)"
3530 GO TO 3035
9000 REM GRAFICOS
9010 FOR n=USR "a" TO USR "k"+7
9020 READ dato
9030 POKE n,dato
9040 NEXT n
9050 DATA 0,0,0,0,0,0,0,1
9060 DATA 0,0,0,0,0,0,0,128
9070 DATA 1,0,0,0,0,0,0,0
9080 DATA 128,0,0,0,0,0,0,0
9090 DATA 0,0,0,24,24,0,0,0
9100 DATA 0,0,0,24,24,0,0,1
9110 DATA 128,0,0,24,24,0,0,0
9120 DATA 0,0,0,24,24,0,0,128
9130 DATA 1,0,0,24,24,0,0,0
9140 DATA 0,0,0,24,24,0,0,24
9150 DATA 24,0,0,24,24,0,0,0
9160 RETURN
9990 SAVE "CRAPS" LINE 5: SAVE "
chars"CODE 64250,768
9991 CLS : PRINT "Programa graba
do"''"Rebobine la cinta y pulse"
'"una tecla para verificar": PAU
SE 0
9992 PRINT '"Pulse PLAY": VERIFY
 "CRAPS": VERIFY "chars"CODE 642
50,768
9993 PRINT ''"Correcto"
```



APARTADO DE CORREOS

Dirige tus cartas a:
Todospectrum
Bravo Murillo, 377, 5.º-A
28020 Madrid

PROBLEMAS CON EL 128

Hace escasamente unos días que he adquirido un SPECTUM 128 y me gustaría que me aclaraseis algunas dudas que se me han planteado:

1. Cuando conecto el interface del joystick y el joystick me pasa del mensaje Copyright al modo mayúsculas, me aparecen unos ceros intermitentemente. Cuando hago Reset me sucede lo mismo. ¿A qué es debido esto?

2. ¿Es perjudicial para el ordenador quitarle el interface del joystick cuando está conectado a la red eléctrica?

3. ¿Que es un Wafadrive?

4. ¿Se le pueden conectar al 128K, directamente las unidades de microdrive?

José Vicente Cheto
Castellón

1. Si el ordenador le funciona correctamente con el interface del joystick desconectado, indudablemente su interface está defectuoso. Póngase en contacto con el vendedor para que se lo repare o cambie por otro.

2. Nunca se debe desconectar ningún tipo de periférico con el ordenador conectado a la red eléctrica, lo más probable es que se deteriore alguno de los dos aparatos.

3. El Wafadrive es un sistema de almacenamiento rápido de datos en cartuchos de cinta sin fin similar en funcionamiento al microdrive de Sinclair.

4. El 128K es totalmente compatible con el Spectrum normal por lo que no debería haber ningún problema en conectar las unidades de microdrive a su ordenador.

APLICACIONES Y FUNCIONAMIENTO DEL RS-232

Soy un asiduo lector de vuestra revista. Me dirijo a vosotros para pediros información sobre el interface RS-232. Desearía conocer cuál es su funcionamiento y las posibilidades que existen de comunicarse con otro ordeandor que no sea Spectrum.

Miguel Simo
Palma de Mallorca

Explicar en esta sección el funcionamiento del RS-232 resulta materialmente imposible debido a la complejidad del tema. Sin embargo, podemos aconsejarte la lectura del «Libro del RS-232» de Anaya Multimedia en el que encontrarás todos los datos que te sean necesarios.

Al ser el RS-232 una norma standard, con el software adecuado podrás conectar el Spectrum a cualquier otro ordenador que disponga de salida RS-232.

EN BUSCA DEL LISTADO PERDIDO

En el número 20 de su revista, en el artículo «La guía del Hacker» falta el programa 2. Les agradecería que resolviesen el problema publicando el listado de dicho programa.

Angel J. Valiente
Castellón

De nuevo los duendes han hecho travesuras con nuestros listados. A continuación publicamos el listado perdido.

```
10 CLEAR 40000:LOAD""CODE:
POKE 65440,81:POKE 65441,101:
RANDOMIZE USR 65458
```

CODIGO MAQUINA DEL 68008

Soy un asiduo lector de vuestra revista, especialmente del suplemento QL. Soy propietario de un QL versión MGE y quisiera que me respondieseis a unas preguntas:

1. ¿Existe algún libro en castellano sobre código máquina del 68008?

2. ¿Cómo puedo conectar una impresora Seikosha SP100 con salida centronics?

Miguel Angel Jiménez
Madrid

1. Existen varios libros sobre el 68000 y 68008 en castellano. Entre ellos se encuentran: «Microprocesadores de 16 bits», (edit. Paraninfo), «68000 guía del usuario» y «QL guía avanzada del usuario» (edit. RAMA), aunque este último trata muy superficialmente el microprocesador dedicándose en su mayoría a la descripción del QDOS.

2. Se encuentra en el mercado un adaptador serie-paralelo que permite conectar el QL con cualquier impresora centronics.

SISTEMAS MIDI PARA SPECTRUM

Ante todo deseo felicitaros por la magnífica revista que publicáis. Agradecería que me facilitáseis direcciones de empresas qwue comercialicen sistemas MIDI para Spectrum o Commodore 64.

Víctor Ordóñez
Gijón

Puedes ponerte en contacto con PIN-SOFT. Paseo de Gracia, 11. 08007 Barcelona; o con Ventamatic. Córcega, 89. 08029 Barcelona.

ENSEÑAR CASTELLANO A UN QL INGLES

En un reciente viaje he adquirido un QL inglés; deseando adaptarlo al castellano, lo llevé al servicio técnico oficial de Sinclair, pero al no disponer de la garantía oficial no me han adaptado el ordenador. Quisiera saber si existe algún otro lugar donde pueda hacer dicha conversión.

Javier Luis Puertas
Madrid

No conocemos ningún establecimiento donde le puedan hacer dicha adaptación. Sin embargo, podemos informarle de la existencia de una tarjeta convertidora que se conecta directamente al bus de expansión. El único inconveniente es que no podrá conectar otro periférico (controlador de discos) simultáneamente. Esta tarjeta la fabrica Abaco, pudiéndose adquirir en las tiendas especializadas por un precio aproximado de 12.000 ptas.

ARANA FOREVER

Mi carta, como otras muchas que os han escrito, se centra sobre el Operativo Arana (aprovecho para felicitar a Manuel por su programa).

Mis conocimientos de Ensamblador son casi inexistentes, pero me gustaría modificar la rutina de impresión de pantalla para hacerlo en 51 columnas. Mi intención no es conseguir listados y mensajes de error de esta forma, sino simplemente la escritura en pantalla. ¿Qué modificaciones (si pueden hacerse) debo efectuar? Según creo, esta parte del programa se publicó en el número de enero y empieza en la línea 1761 (PROUT).

Raúl Lión
Valladolid

El conseguir que nuestro Spectrum imprima a 51 columnas en lugar de sus 32 standard es más difícil aún que el convencerlo de que lo haga a 64. No es que sea imposible, pero pensamos que no vale la pena el esfuerzo, ya que las aplicaciones para las que puede servirnos son demasiado específicas como para que resulte de interés general.

Aprovechamos el momento para destacar que no es esta una sección de programas «a medida», sino un espacio dentro de TODOSPECTRUM en la que tienen cabida vuestras opiniones, preguntas y comentarios sobre todo tipo de temas. Viene esto a cuento porque, de un tiempo a esta parte, parece que estas líneas venían siendo utilizadas exclusivamente como un último recurso a la hora de buscar el modo de resolver vuestros problemas de carácter técnico. Y no es que no esté bien que nos comuniquéis vuestros problemas y dudas por si podemos seros útiles, pero empezamos a cansarnos de que nos pongáis en un compromiso cuando, por ejemplo, nos pedís la forma de que funcione cierto programa comercial con un determinado tipo de impresora e interface (de los que, a lo peor, no tenemos suficiente información técnica) mientras los responsables de esta falta de información que a todos nos afecta (sean fabricantes, distribuidores o vendedores) duermen tranquilos.

Nos gustaría que esta sección fuera algo mucho más abierta y más humana también; una vía de comunicación lectores-redacción por la que pasarán todas vuestras inquietudes y gracias a la que pudierais ver publicada cualquier coda que os apetezca. Esperamos una verdadera lluvia de cartas en las próximas semanas. Mientras tanto, amigo Raúl, te recomendamos que sigas con asiduidad la serie de Lenguaje Máquina para que puedas resolver por ti mismo ese problema que tanto te preocupa.

QL SUPERCHARGE, EL COMPILADOR FANTASMA

Soy suscriptor de su revista TODOSPECTRUM, y estoy muy satisfecho de ella, sobre todo la sección dedicada al ordenador QL.

En el número de abril del 86 hubo un artículo dedicado al QL en que se mencionaba el compilador de Digital Precisión SUPERCHARGE. Como me interesa lo quise comprar y no me fue posible ya que no sabían que existía. ¿Me pueden dar alguna dirección para adquirirlo?

Luis de la Rosa
Barcelona

El programa al que hace mención fue adquirido en una de nuestras visitas al Reino Unido, y no sabemos con certeza si en estos momentos alguna casa ha iniciado su distribución por nuestro país o si esto llegará a hacerse algún día. Todo lo que podemos ofrecerte por el momento son las señas del comercio donde lo adquirimos por si pudiera negociar su adquisición por correo. Estas son:

SONIC FOTO CENTER
256 TOTTENHAM COURT ROAD
LONDON W1P 9AD
Tel.: 01-580 5826/7

¡Ojo!, el precio que tuvimos que pagar por el programa rozó las 100 Libras esterlinas.

EL CORCHO



Vendo Spectrum 48 K, cables, alimentador, manual (en español y en inglés), revistas, más de 40 juegos, interface de joystick Kempston y el libro «Profundizando en el Spectrum». Todo en buen stado por 23.000 ptas. Interesados escribir a Juan J. Sosa Lorenzo. Pza. del Escorial, 4, 5.°. Las Plamas de Gran Canaria. Tel.: (928) 36 87 03.

Club para principiantes del Spectrum. Intercambios. Escribid a J. H. A. Federico García Lorca, 21, 8.° B. Algeciras (Cádiz).

Vendo Spectrum 48 K con teclado multifunción 1, accesorios y documentación, junto con numerosos programas de juegos y utilidades, entre ellos lenguajes Logo castellano, Forth y Pascal de Hisoft, por sólo 30.000 ptas. Llamar noches al Tel.: (91) 315 71 93. Preguntar por Tomás.

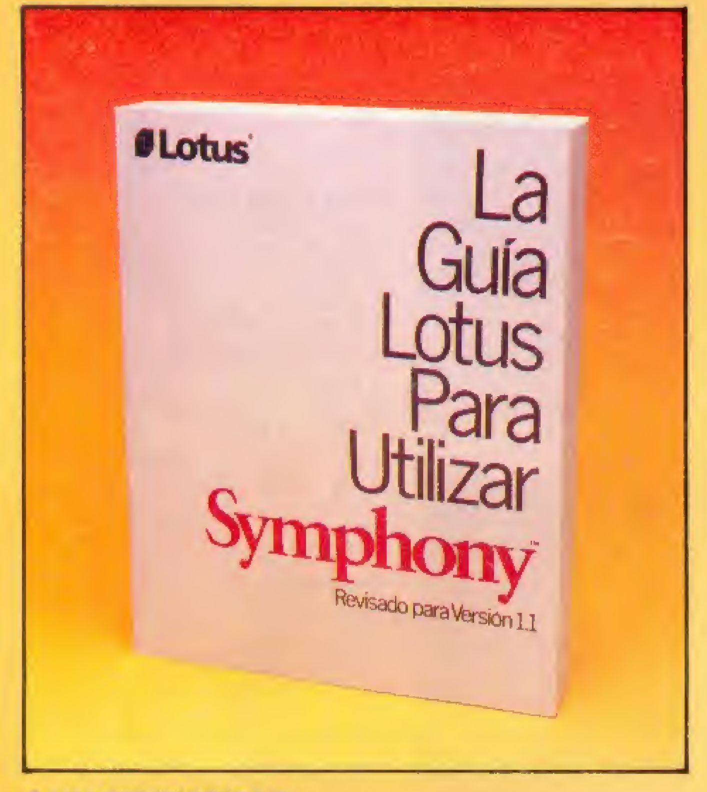
Vendo ordenador Spectrum 48 K, juegos, programas y manual, Interface 1, microdrive, Interface Indescomp serie y paralelo. Precio 35.000 ptas. negociables. También los vendemos por separado, precio a convenir. Ana. Tel.: (91) 437 49 03.

Vendo Interface 1, 2 unidades de microdrive, 20 cartuchos de microdrive, libro de manejo y manual de instruccions. Todo en 25.000 ptas. Joaquín Bayón López. Capitán Almeida, 28, 1.° B. Tel.: (985) 22 61 13. 33009 Oviedo.

Vendo programas para Casio PB 100, PB 110, PB 700, PB 770, FX 750-P. Envío lista completa. Luis Alonso Pablo, junior. José Ricart, 44, At. 3.ª. Sant Feliu de Llobregat. 08980 Barcelona. Tel.: (93) 666 02 27.

Lotus®

La Guía Lotus Para Utilizar Symphony



CARACTERISTICAS:
- Páginas: 443
- Papel offset: 112 grs.
- Tamaño: 182 x 232 mm.
- Encuadernación: Rústica-cosido

LA GUIA LOTUS PARA UTILIZAR SYMPHONY es un libro que le enseñará paso a paso, y de una forma muy práctica cómo utilizar este programa.

LA GUIA LOTUS contiene:
- Cómo crear y manejar ficheros
- Descripción detallada de las facilidades que ofrecen las ventanas de SYMPHONY.
- Apéndice que cubre las aplicaciones adicionales que van incluidas en el programa.
- Un índice detallado y un vocabulario donde fácilmente podrá encontrar cualquier tema que necesite.

El complemento indispensable para el manual de **SYMPHONY**

OFERTA DE LANZAMIENTO 4.500 PTAS. (IVA INCLUIDO)

Recorte y envíe HOY MISMO este cupón a: **infodis, s.a.** c/ Bravo Murillo, 377 - 28020 MADRID

CUPON DE PEDIDO

SI. Envíenme el libro «**LA GUIA LOTUS PARA UTILIZAR SYMPHONY**» al precio de **4.500** PTAS. EL IMPORTE lo abonaré:

Con tarjeta de crédito VISA ☐ INTERBANK ☐ AMERICAN EXPRESS ☐
CONTRAREEMBOLSO ☐ ADJUNTO CHEQUE ☐

Número de mi tarjeta ____________________
Fecha de caducidad ____________ Firma, ____________
NOMBRE ____________________
DIRECCION ____________________
CIUDAD ____________ C.P. ________
PROVINCIA ____________ TELEFONO ________

TAMBIEN LO PUEDE ADQUIRIR EN SU LIBRERIA HABITUAL